Cultura humanizada no meio tech

Estratégias para pessoas e organizações construírem ambientes mais saudáveis

Casa do Código | alura

ANDRÉ GUIMARÃES SAKATA

Sumário

- ISBN
- Agradecimentos
- Prefácio
- Sobre o autor
- Estamos esgotados
- PARTE 1 — O TIME E SUAS FORMAS DE TRABALHAR
  - 1 O que faz a pessoa desenvolvedora de software?
  - 2 Estilos de gestão e suas consequências
  - 3 Segurança para assumir riscos e estar vulnerável
  - 4 Desenvolvimento centrado no ser humano
- PARTE 2 - A EMPRESA E SUA CULTURA
  - 5 Por que o trabalho está uma loucura?
  - 6 Conversas difíceis
  - 7 Lidando com falhas
- PARTE 3 - O INDIVÍDUO E SUAS CRENÇAS
  - 8 Padrões de pensamentos
  - 9 Gestão de conhecimento, tempo e energia
- CONSIDERAÇÕES FINAIS
  - 10 Assumindo nossa responsabilidade
  - 11 Construindo um plano de ação

ISBN

Impresso: 978-85-5519-323-1

Digital: 978-85-5519-322-4

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

Agradecimentos

Durante toda a minha carreira, tive a sorte de ter encontrado pessoas incríveis que colaboraram imensamente para que eu construísse (e para que eu continue construindo) uma relação saudável com minha profissão. Em especial, sou muito grato pelas pessoas que conheci e com quem colaborei no desenvolvimento da Vitalk (serviço de saúde mental para empresas). Essa passagem foi o que me instigou a olhar para cultura organizacional e saúde mental de uma outra maneira dentro do desenvolvimento de software.

Além disso, especificamente para a elaboração deste material, algumas pessoas precisam ser destacadas. A idealização desse projeto surgiu de conversas que tive com um amigo de longa data, Alan Fachini. Muitas referências que há neste livro me foram apresentadas por ele e várias ideias surgiram de conversas que tivemos sobre nossas experiências como engenheiros de software e gestores.

A jornada de exploração para o desenvolvimento deste livro me levou a navegar sobre assuntos diversos e a editora deste trabalho, Vivian Matsui, contribuiu enormemente para que eu fosse capaz de organizar e sintetizar esses conteúdos de uma maneira prática e lógica. Sem ela, o resultado final provavelmente seria muito mais caótico e difícil de assimilar.

Uma grande influência para minhas pesquisas foi o fundador do Software Zen, Alisson Vale. Realizei alguns de seus cursos durante a elaboração deste livro e seus conteúdos sempre me fizeram refletir profundamente sobre o sentido de muitos assuntos que tangem o desenvolvimento e gestão de projetos de software. Muitos *insights* contidos aqui surgiram de sínteses de seus materiais e aulas.

Por fim, uma boa parte do conteúdo deste livro foi escrito durante a pandemia de COVID-19. A principal pessoa com quem conversei sobre o teor das ideias descritas aqui, e que me incentivou a ir em

frente nesse trabalho, foi minha noiva, Idineide Viana. Foi uma sorte imensa ter sua companhia, assim como de meus familiares, nesse período turbulento e cheio de incertezas. Suas presenças preenchem minha vida de sentido.

Prefácio

O livro que você tem em mãos começa com um anúncio muito dito, mas pouco ouvido: estamos esgotados! Sim, eu já estive. Talvez você também já tenha estado, ou pode até estar neste exato momento. A humanidade está esgotada. Em especial, líderes e desenvolvedores em projetos de software têm sua saúde mental deteriorada por décadas sem sequer perceberem o que está acontecendo à sua volta.

A primeira manifestação desse esgotamento surgiu ainda na década de 60. Foi em uma conferência promovida pela OTAN em 1968. Ali, os engenheiros que presenciavam o início da nossa era digital já se preocupavam com dificuldades crônicas para o atingimento de prazos e para o desenvolvimento de funcionalidades em projetos, cujo grau de complexidade aumentava progressivamente. Já naquela época, as dificuldades de estimar e planejar o esforço e o trabalho envolvidos para conclusão de um projeto de tecnologia eram grandes, o que resultou em pressão de entrega, horas extras e insatisfação generalizada. Nas décadas seguintes, intensificou-se a constatação de que conduzir um projeto de software não era como gerir um projeto comum.

À medida que os computadores se tornavam exponencialmente mais poderosos e que se ampliava a infraestrutura de software e comunicação, mais potencialidades de uso e de aplicação eram disponibilizadas para o público em geral. Para as empresas, essas potencialidades se manifestavam em termos de aumento de eficiência, de ampliação e construção de novos negócios e na necessidade de mais investimentos. Quanto mais investimentos, mais projetos, mais pessoas envolvidas e dificuldades para o sucesso.

Apesar de imensa, essa evolução técnica não foi acompanhada por uma gestão eficaz dos projetos na ponta final das empresas. Usamos o que já tínhamos à mão, ou seja, conceitos e técnicas de

gestão genéricas que desconsideravam as características intrínsecas do desenvolvimento de software. Isso ampliou o problema da ineficácia e, principalmente, do impacto negativo na sustentabilidade e saúde mental dos profissionais envolvidos. A "Crise do Software", como ficou conhecido esse período da nossa indústria, ajudou a estabelecer uma cultura de heroísmo, na qual somente o profissional que se dedicasse acima da normalidade era capaz de vencer projetos malsucedidos.

Na virada dos anos 2000, desenvolvedores em várias partes do mundo, cientes e experienciadores frustrados dessas dificuldades, começaram a reconhecer as características técnicas e sociais que aumentavam a probabilidade de bons resultados em projetos de software. Não foi nenhuma surpresa constatar que o foco na forma como as pessoas envolvidas se relacionavam e na sustentabilidade técnica dos processos tornou-se seus princípios orientadores.

A atenção passou agora a se concentrar nas condições técnicas e sociais que traziam mais eficácia para os projetos, ou seja, que entregavam a coisa certa na hora certa. Em vez de pesados processos que tentavam controlar cada atividade para vencer o medo e diminuir o risco, o jogo nesse momento passou a invocar o desejo da vitória em vez do medo da derrota. Com isso, aceitamos incorrer em novos riscos – e isso fez toda a diferença.

A natureza criativa e colaborativa do desenvolvimento de produtos começou a ser resgatada, assim como uma atenção especial ao valor que precisava ser agregado. Um cuidado maior passou a ser dado ao relacionamento colaborativo entre as pessoas envolvidas. Excelência técnica passou a ser ao mesmo tempo requerida e desejada para que os produtos crescessem de forma sustentável. Os cuidados com o humano e com o técnico se uniram para reposicionar os modelos de gestão que eram usados até então. Isso foi, e ainda continua sendo, a essência do movimento Ágil na indústria de software.

É orientado por esses princípios que este livro articula as soluções disponíveis no momento para o problema da questão da saúde

mental e da sustentabilidade de processos no mundo contemporâneo. Embora tenhamos virado o barco para a direção certa, e apesar de muitos dos problemas iniciais terem sido endereçados, a condição atual de profissionais em um projeto de software ainda deixa a desejar.

Cada vez mais conhecemos empresas com culturas que colocam essa preocupação no centro das atenções. Mas o jogo mais comum ainda continua sendo disfuncional: trabalho empurrado; prazos arbitrários; incompreensão do valor que precisa ser agregado; relações tóxicas. O trabalho continua se expandindo para absorver todas as horas das nossas vidas, deixando de lado a criatividade e o significado que nos levaria a apreciá-lo.

Essa cultura nos faz acreditar, ou aceitar, que o resultado das nossas ações vem a partir do volume de trabalho que realizamos, do medo, da competição, do controle ou da eficiência para se manter vários pratos girando simultaneamente. Muitas vezes, senão sempre, está aí a origem de tantos ambientes de trabalho disfuncionais.

O problema é que essas condições às vezes “dão resultado”, especialmente quando incêndios precisam ser apagados. Esse é um grande problema, porque o suposto "bom resultado" no curto prazo esconde o real impacto de se manter tal comportamento no longo prazo. O choque de realidade vai em algum momento se manifestar na nossa vida pessoal – na forma de sofrimento, divórcios, distanciamento dos filhos, doenças físicas e psicológicas.

Além disso, há o efeito do próprio estilo de condicionamento que imputamos a nós mesmos. Começamos a formar hábitos difíceis de serem desconstruídos, como empurrar trabalho para horas extras, fins de semana ou feriados; não saber dizer não; não saber selecionar a coisa certa a se fazer a cada momento; ou não saber separar o essencial do prescindível. A criatividade, indispensável para se fazer melhor, é desfavorecida. O jogo passa a ser apenas “fazer mais”; só a eficiência passa a ter valor. O jogo se torna maquinal e gerador de sofrimento.

Quando penetram na organização, esses hábitos e comportamentos que desviam a orientação humana para o plano da eficiência assumem novos contornos. Vemos o mesmo padrão se reconfigurar na organização: por meio da realização de planejamentos longos e detalhados; do "Big Design Up Front"; do excesso de trabalho em progresso; das tentativas elaboradas de "paralelizar" projetos e outros itens de trabalho; das medições de performance e produtividade individual; do desenvolvimento de soluções para problemas que não existem ou que não são importantes ou prioritários; das relações tóxicas. Nesse ímpeto por "fazer mais", indivíduos e organizações alimentam-se uns aos outros, produzindo uma espiral autodestrutiva no médio e longo prazo.

O problema não é simples e este livro oferece três importantes perspectivas para nos ajudar a endereçar essa questão no dia a dia dos nossos projetos. Uma perspectiva técnica, que diz respeito à forma como os times se formam e como eles agem; uma perspectiva cultural, voltada para a empresa e as crenças que sustentam sua forma de operar; e, claro, uma perspectiva individual, relativa ao autoconhecimento da pessoa participante no esforço coletivo, aquela para o qual todo o sofrimento se volta.

Quando esse quadro mais abrangente é bem compreendido, uma virada se torna possível de acontecer: tanto a saúde mental quanto a sustentabilidade do nosso processo de trabalho se tornam muito mais premissas a serem seguidas do que um alvo a ser buscado.

A experiência no trabalho passa a vir primeiro, deixando que o resultado emerja a partir de um berço mais acolhedor e construtivo. Será que estamos preparados para dar esse passo? Cada vez mais. O texto conciso, prático e abrangente do André Sakata vai certamente ajudar o leitor que já está pronto progredir nessa direção.

Alisson Vale

Agosto, 2022

Sobre o autor

André Guimarães Sakata foi o CTO e cofundador da Vitalk, startup que surgiu com a missão de tornar a saúde mental mais acessível. A empresa foi adquirida pela Gympass no início de 2022 e desde então trabalha como Staff Software Engineer na companhia. Durante sua carreira, trabalhou como engenheiro de software, líder técnico e empreendedor de diversas iniciativas e participou como palestrante em eventos e conferências de tecnologia. Pode ser encontrado no LinkedIn (https://www.linkedin.com/in/andresakata/) e no GitHub (https://github.com/andresakata).

Estamos esgotados

Desenvolver software é incrível. Quando fizemos nosso primeiro "hello world", sentimos um potencial enorme. Percebemos que somos capazes de construir coisas, de logicamente organizar ideias e, em pouco tempo, produzir algo novo — e isso é muito prazeroso. Além disso, ver esses produtos serem distribuídos de forma instantânea e transformarem o comportamento das pessoas é viciante.

Mas, de alguma forma, esse cenário se torna um inferno.

De alguma forma, toda essa sensação de potencial e liberdade que sentimos quando fizemos nosso primeiro software funcionar se transforma em sofrimento. Quando chegamos ao mundo real, nós nos deparamos com projetos diferentes do que costumávamos fazer. Esses projetos possuem nuances. Eles parecem estar claros em um momento e ficam nebulosos no momento seguinte. Ficamos ansiosos com suas incertezas e trabalhamos arduamente para fazer progresso, mas parece que nada é suficiente. Tentamos inúmeras vezes e não encontramos o jeito certo de implementá-los. Sentimos vontade de refazer tudo, de começar do zero, mas isso não é possível.

Por fim, sentimos culpa por não estarmos preparados o suficiente.

Esse trabalho é uma humilde tentativa de colocar luz sobre vários desafios que enfrentamos como pessoas inseridas na indústria de software. Gostaria de articular uma compreensão mais ampla sobre a natureza do nosso trabalho, no intuito de que nós e nossos times mantenhamos uma boa saúde mental dentro de cultura empresarial sustentável.

A primeira vez que eu comecei a pensar com mais profundidade sobre sustentabilidade humana em desenvolvimento de software foi quando eu assisti a uma palestra na RubyConf de 2019, em São

Paulo. Em uma das trilhas paralelas, havia uma apresentação sobre saúde mental e fiquei curioso. A apresentação era basicamente um relato sobre como um desenvolvedor passou por um *burnout*, seguido de uma crise de depressão.

Eu não consigo mais me lembrar dos detalhes da história, mas eu me lembro como me senti. Lembro que fiquei profundamente tocado porque conseguia entender claramente como um problema levou a outro e como a relação com o trabalho arruinou, por um momento, sua própria vida. Eu podia ver que aquilo poderia ter acontecido comigo.

Aquele não era apenas um acontecimento isolado nem específico do mercado de desenvolvimento de software. Para começar, vamos falar da dimensão econômica: estima-se que o estresse ocupacional custe mais de 300 bilhões de dólares nos Estados Unidos em absenteísmo, baixa performance e custos com saúde[1]. Segundo publicação do *National Institute for Occupational Safety and Health*, problemas de trabalho possuem uma correlação mais forte com queixas de saúde do que qualquer outro fator estressor, inclusive problemas financeiros e familiares[2].

Apesar de os números agregados serem alarmantes, o que realmente choca são as histórias particulares. Em 2017, Joseph Thomas, engenheiro de sofware da Uber, deixou sua esposa e dois filhos ao se suicidar. Apesar do alto salário pago pela empresa, sua esposa Zecole viu sua vida desmoronar com a cultura agressiva e tóxica do novo emprego. Thomas era um profissional competente, recusou um emprego na Apple e deixou um cargo no LinkedIn para trabalhar na Uber e, apesar disso, teve sua autoestima arruinada por seus superiores. Ele não acreditava mais em si mesmo; chegou a receber a recomendação de sua terapeuta para deixar a companhia, mas não conseguiu fazer isso a tempo.

Quando eu ingressei no mercado de trabalho, lembro de ter ouvido a história de um gerente de projetos que era excepcional. Era uma pessoa jovem ganhando um salário alto. Quando atingia metas,

ganhava bônus que incluíam viagens ao exterior. Ao mesmo tempo, ele engordara mais de 20 quilos em um ano e já começava a ter a saúde comprometida. Apesar de tudo, essas consequências eram vistas como uma medalha de honra. Esse seria o preço a pagar caso queiramos obter sucesso.

O estresse crônico oriundo do trabalho leva a problemas em todas as esferas da vida. Os relacionamentos vão se arruinar porque é impossível cultivar qualquer relação de qualidade quando não há mais nenhuma energia no final do dia. Os hobbies que costumávamos ter, cuja atividade proporcionava o prazer em si e não em um resultado extrínsico, param de fazer sentido. O propósito que tínhamos quando começamos a trabalhar começa a ficar mais distante e cinzento. Por fim, nossa saúde física começa a se deteriorar quando nos afastamos de bons hábitos e começamos a adotar vícios nocivos.

A importância da saúde mental para os indivíduos parece óbvia. Pessoas literalmente morrem por seus salários.

No entanto, quando pensamos do ponto de vista das empresas, podemos imaginar que há um conflito. Se as pessoas estão esgotadas no trabalho, ao menos elas devem estar produzindo mais e gerando lucro. Mas não estão. O custo desse estresse e esgotamento não é recompensado com mais resultados, porque o trabalho criativo não pode ser exprimido com força bruta. O ambiente tóxico para os indivíduos resulta em baixo engajamento e rotatividade, e reduz a produtividade das empresas.

Isso nem sempre acontece apenas por sobrecarga de demandas. A poluição social, que surge de relações tóxicas, leva ao cinismo e consome a energia das pessoas com o desgaste social. Dentro do desenvolvimento de software, onde existe uma forte interdependência entre os trabalhadores, como entregar algo se você não suporta mais a convivência?

Independente dos resultados econômicos, deveríamos pensar sobre a sustentabilidade humana de nossas organizações porque essa é, simplesmente, a coisa certa a se fazer. É uma questão de ética e moral. As empresas deveriam não apenas permitir que seus funcionários tenham tempo de qualidade após o expediente, mas incentivar que isso aconteça, por nenhum outro resultado além disso.

Com isso, as reflexões e propostas deste livro serão divididas em três partes:

- O time e suas formas de trabalhar;
- A empresa e sua cultura;
- O indivíduo e suas crenças.

Na primeira parte, revisitaremos vários conceitos bem estabelecidos nas últimas décadas em gestão de projetos de software, sob uma perspectiva centrada nas necessidades humanas atendidas por essas metodologias. Além da busca por eficiência e eficácia, de que forma o movimento Ágil contemplou as necessidades humanas por trás do seu manifesto e o que pode estar passando despercebido? Nessa parte, estaremos no escopo das equipes e como elas trabalham.

Na segunda parte, faremos reflexões voltadas as crenças que estão presentes nas organizações como um todo. Pensaremos sobre como essas crenças podem ser nocivas aos objetivos da companhia e da sustentabilidade humana de seu pessoal e o que podemos fazer a respeito disso.

Na última parte, entraremos no campo individual. Apesar dos pontos de vista técnicos e culturais da primeira e segunda parte, boa parcela de nosso sofrimento tem origem em nós mesmos. Nossos padrões de pensamentos possuem grande influência sobre nossas ações e suas consequências. Falaremos sobre o *burnout* sob uma perspectiva individual e apresentaremos abordagens para gestão de energia. Embora muitas vezes nos encontremos em ambientes

negativos, podemos ser agentes de transformação a partir de nós mesmos.

É evidente que cada caso de *burnout* é singular. Ainda que sejam apresentadas dicas práticas sobre como lidar com algumas situações, este não é um manual de como resolver o *burnout* dentro do desenvolvimento de software.

Entretanto, este livro é para todas as pessoas que acreditam que é nosso dever criar um ambiente em que pessoas possam trabalhar por anos sem necessariamente enfrentar doenças resultantes de estilos desumanizados de gestão — e que podemos fazer isso ao repaginar nossas metodologias e crenças pessoais.

Buscaremos compreender melhor a natureza do nosso trabalho para recuperar (ou não perder) nosso prazer em desenvolver software e colocar nossa atividade profissional como fator construtor da nossa saúde, ao invés de origem de doenças.

Referências

[1] PFEFFER, Jeffrey. *Morrendo por um salário*: como as práticas modernas de gerenciamento prejudicam a saúde dos trabalhadores e o desempenho da empresa — e o que podemos fazer a respeito. Rio de Janeiro: Alta Books, 2019.

[2] National Institute for Occupational Safety and Health. *Stress... at work*. U.S. Department of Health and Human Services. Centers for Disease Control and Prevention. Publication n. 99-101. Disponível em: (https://www.cdc.gov/niosh/docs/99-101/default.html). Acesso em 16 set. 2022.

PARTE 1 — O TIME E SUAS FORMAS DE TRABALHAR

Capítulo 1
O que faz a pessoa desenvolvedora de software?

Estamos esgotados. Mas como chegamos até aqui? Para responder a essa pergunta, precisamos antes responder a outra. Afinal, quais são as atribuições e responsabilidades de desenvolvedores de software?

De forma simplista, podemos dizer que programadores produzem interfaces, algoritmos, estruturas de dados e códigos em geral. Eles recebem um conjunto de requisitos e fazem uma reprodução fiel do produto. Nessa visão, os melhores programadores são os que realizam o trabalho mais rapidamente e com menos divergência sobre o projeto, ou seja, com menos *bugs*.

Essa resposta vem da abordagem analítica que herdamos da Revolução Industrial. Com esse olhar, o trabalho é sempre empurrado para etapas seguintes, como uma fábrica. Por exemplo: uma demanda é criada por uma gerente e entregue a um analista; a especificação é gerada por um analista e entregue a um designer; as interfaces são construídas por um designer e entregues a uma programadora; o software é criado por uma programadora e entregue a um testador; a aprovação é realizada pelo testador e o software é entregue ao cliente. Quando a demanda chega à última etapa, ela está concluída e, quando todas as demandas de um projeto chegam à última etapa, o projeto está concluído. Na prática, buscamos ganhar escala ao dividir as demandas em partes menores e entregá-las para pessoas com funções bem delimitadas.

Mas há uma diferença entre programar e desenvolver software. Quando estamos aprendendo uma nova linguagem, fazendo uma

prova na universidade ou resolvendo um desafio de programação, essas atividades são bem diferentes de outras, como alterar um software com usuários, desenvolver um negócio ou alterar um produto que faz parte de algo maior, como uma organização.

Quando estamos apenas programando, o problema não é ambíguo. O software esperado como resposta para uma prova é objetivo e bem delimitado. Nesse caso, a atividade pode ser isolada porque o software é o objetivo em si mesmo.

No caso de desenvolver software, o cenário fica mais caótico. O problema é ambíguo e há infinitas maneiras certas e erradas para buscar resultados, que também podem estar certos e errados. Nesses casos, o ambiente não se mantém estável. Os fatores do sistema (seja o software, ou a organização como um todo) mudam bruscamente. Desenvolvedores estão inseridos dentro de uma equipe multidisciplinar porque o software é um meio para se atingir um objetivo que está fora dele.

Programar	Desenvolver software
Exemplos: aprender uma nova linguagem; fazer uma prova na universidade; resolver um desafio de programação.	Exemplos: desenvolver um negócio; alterar um software com usuários; implementar um software que faz parte de algo maior.
Problema não ambíguo; objetivo.	Problema ambíguo; subjetivo.
Atividade solitária, ou com outros programadores.	Atividade em equipe multidisciplinar.
O software é o objetivo em si.	O software é um meio para atingir um objetivo que está fora dele.

É evidente que a abordagem de Taylor para a gestão de projetos de software não funciona. Ela não passa pelo teste da realidade. Com alguma experiência, aprendemos que alguns requisitos vão emergir durante o desenvolvimento, que não poderiam ser descobertos *a priori*, e isso impede que uma abordagem em cascata funcione para a categoria de problemas que enfrentamos.

Mesmo assim, hoje em dia muitas empresas ainda trabalham nesse modelo, deixando seus profissionais criativos desconectados com o propósito de seu trabalho e transformando-os em "entregadores de tarefas". Em muitos casos, as organizações nem se dão conta de que estão trabalhando dessa forma, porque há práticas ágeis que mascaram o esquema de trabalho. Eles possuem um projeto de vários meses estabelecido e negociado *a priori*, mas as partes estão em um quadro Kanban e, por causa disso, pensam que estão sendo ágeis.

Os motivos para as abordagens puramente analíticas se perpetuarem nas empresas para gerenciar software podem ser vários. Em primeiro lugar, elas de fato podem dar certo em alguns

casos. Analisar um problema e dividi-lo sempre será parte fundamental do processo de entregar software. Aqui, não proponho o abandono de toda a análise. O problema acontece quando não se enxergam as hipóteses e premissas por trás da delimitação. Nesse caso, o surgimento das demandas durante a implementação, comuns no processo de descoberta, passa a ser considerado uma exceção e um erro de especificação e, de repente, a empresa se vê no cenário em que muito foi construído, mas não há nada pronto que entregue valor.

Em segundo lugar, as empresas querem ter controle e previsibilidade, mesmo sobre um trabalho que é naturalmente inovador e incerto. Entretanto, se estamos realmente preocupados com os resultados que almejamos alcançar através da implementação de um software, não podemos desconsiderar a possibilidade de que houve equívocos na definição do projeto. Por falta de conhecimento e pressão, gerentes acabam respondendo a incertezas com mais controle.

> "*Quanto mais eficiente você for em fazer a coisa errada, mais errado você se torna. É muito melhor fazer a coisa certa incorretamente do que a coisa errada corretamente. Se você fizer a coisa certa da maneira errada e corrigi-la, você se torna melhor."* – Russel L. Ackoff [1]

De fato, as empresas possuem restrições (geralmente financeiras) e consequentemente possuem necessidade de ter previsibilidade sobre seus projetos. Ao mesmo tempo, há incertezas fundamentais do processo de inovação, e não se pode ter garantias sobre os resultados almejados do desenvolvimento de um projeto. A partir desse impasse, e com a falta de habilidade de lidar com ele, as relações dentro das organizações começam a se deteriorar.

Desenvolvedores se sentem pressionados a aceitarem acordos irrealistas e começam a agir na defensiva. Gerentes de projetos

ficam ansiosos porque não conseguem confiar nos acordos estabelecidos com desenvolvedores, que são quebrados a cada semana, e começam a buscar mais informações, com mais planilhas, quadros e reuniões de check-in. Clientes pensam que o projeto está demorando mais do que deveria e começam a pressionar um rompimento de contrato, porque começam a desconfiar se o software realmente será entregue e poderá ajudá-los a obter os resultados que desejam. De um ambiente com essas relações, podemos esperar o pior para todos os lados.

Porém, há condutas mais eficazes. Metodologias como Kanban e XP, que incorporam ciclos de feedback para buscar fazer a coisa certa, ao invés de produzir software inútil em abundância de forma industrial, passaram a ficar mais populares porque se encaixam de maneira mais apropriada com essa dinâmica. Elas incorporam o pensamento sistêmico, de forma que buscamos compreender o todo com a interação de suas partes, e não as partes isoladas.

Nesse sentido, dado que na maioria das vezes existem um problema nebuloso e uma solução desconhecida, o papel do desenvolvedor ou da desenvolvedora é contribuir para as descobertas durante esses ciclos de aprendizagem. Sua responsabilidade é colaborar e usar seu conhecimento para produzir mais conhecimento.

Interpretar toda a informação fragmentada e organizá-la de forma que seja possível tomar decisões e entregar valor é uma habilidade que toda pessoa programadora precisa aprimorar. Portanto, a figura da pessoa desenvolvedora faz parte do que foi chamado de "trabalho do conhecimento".

1.1 As implicações do "trabalho do conhecimento"

O termo "trabalhador do conhecimento" foi primeiramente descrito por Peter Drucker em seu livro *The Landmarks of Tomorrow* [2], em 1959. Naquela época, ele percebeu que o mundo estava mudando para um sistema econômico em que o conhecimento e a produção de conhecimento seriam mais valiosos que qualquer outro recurso. Nesse contexto, os trabalhadores do conhecimento contribuiriam com suas ideias, passando a ser considerados ativos importantes para as organizações e não mais um custo necessário para uma operação funcionar.

O trabalho criativo sempre existiu, mas a sua valorização na economia é uma mudança sem precedentes. Essas mudanças requerem uma forma diferente de gerir pessoas, porque medir a produtividade dos indivíduos passou a ser muito mais difícil.

Como eu poderia medir a produtividade de uma pessoa que produz conhecimento? O conhecimento não é produzido de forma linear. Quando estamos no meio do processo, não podemos falar com precisão se estamos perto ou longe de chegar aonde queremos.

Da mesma forma, como poderíamos medir a produtividade de uma pessoa programadora? É evidente que a quantidade de linhas de código que ela produz não é uma boa métrica. Linhas de código por si só não representam valor entregue. Funcionalidades que não resolvem nenhum problema são piores do que nenhum código, porque elas de alguma forma reduzem a velocidade da organização. Essas funcionalidades carregam dependências e pesam para a equipe. Além disso, número de linhas não refletem necessariamente a complexidade de uma tarefa.

Quem sabe a média dos *story points* que um(a) programador(a) costuma entregar possa ser uma boa métrica? Com isso, nós poderíamos considerar a complexidade e tamanho das demandas na equação. Ainda assim, será que isso gera os incentivos corretos? Colocar uma meta de *story points* incentiva que os programadores escrevam a maior quantidade de software possível e criem uma negociação delicada nos orçamentos das tarefas.

No fim, esses incentivos ainda desprendem o profissional do resultado do seu trabalho. Dado que seu objetivo é entregar a maior quantidade possível de pontos, ele buscará as tarefas mais confortáveis e não as mais importantes. Além disso, ele não será encorajado a colaborar com a equipe, o que é péssimo para um esquema de trabalho que fundamentalmente possui interdependência.

Medir produtividade de programadores por *story points* possui a premissa de que todo progresso é positivo e retira essa responsabilidade de avaliação do desenvolvedor. Ele não tem de se preocupar com as consequências de seu trabalho.

Quando entramos no tema de mensuração de produtividade individual, normalmente as propostas soam injustas ou contraproducentes. As tentativas de estabelecer metas sobre indicadores tangentes (como quantidade de tarefas concluídas), especialmente quando associadas a bonificações individuais, gerarão interesses contraditórios em algum momento.

Nessa ânsia pela produtividade no estabelecimento de metas, as empresas colocarão seus profissionais em conflito de interesse. Desenvolvedores terão que escolher entre otimizar uma métrica individual ou fazer a coisa certa para caminhar em direção ao objetivo do projeto. Quando as organizações partem da premissa de que as pessoas se motivam exclusivamente pelo interesse próprio, elas criam um esquema de trabalho que força as pessoas a pensarem exclusivamente no interesse próprio — e um círculo vicioso se fortalece.

Isso não significa que métricas não devam ser usadas. Elas claramente possuem seu valor para diagnosticar problemas e identificar gargalos no sistema de trabalho. Porém, o significado e as motivações dessas métricas soam completamente diferentes do que acontece em uma produção em série.

Os problemas gerados pela aplicação da visão analítica e industrial ao gerenciar trabalhadores criativos não foram percebidos apenas recentemente. Peter Drucker percebeu que as características do trabalho criativo conflitavam com as necessárias para uma linha de produção eficiente. Esses profissionais precisavam desenvolver a habilidade de definir suas próprias tarefas, ao invés de seguir um procedimento preestabelecido. Eles deveriam ter autonomia para gerenciar seu próprio tempo, em vez de serem controlados como máquinas. Especialmente, o resultado dos seus trabalhos vem da qualidade de suas escolhas, e não do volume de trabalho entregue.

VUCA

Olhar o desenvolvimento de software sob a ótica do trabalho do conhecimento implica em uma série de consequências. O acrônimo VUCA – que significa *Volatile* (volatilidade), *Uncertainty* (incerteza), *Complexity* (complexidade) e *Ambiguity* (ambiguidade) – tem sido bastante usado para descrever problemas enfrentados por empresas modernas e nós o usaremos para interpretar nossa condição como trabalhadores do conhecimento, com alguns exemplos.

Em um ambiente volátil, mudanças bruscas acontecem. Empresas precisam se transformar com frequência para se adaptarem e sobreviverem. Nesse contexto, desenvolvedores precisam agir com urgência para implementar essas adaptações, que não respeitam a arquitetura de um sistema que foi idealizada para outros casos de uso.

A própria demanda de conhecimento pode mudar de forma volátil e transitar de uma área para outra dentro de uma equipe de desenvolvedores. Em um momento, demandas que exigem conhecimento profundo sobre estrutura de dados e algoritmos podem ser o maior desafio; em outro momento, podem ser necessidades de desenvolvimento de aplicativos e interfaces. As empresas precisam atravessar as mudanças com leveza para resistirem e prosperarem.

Podemos olhar o atributo da incerteza como causa da volatilidade. Como não sabemos se as hipóteses apostadas no desenvolvimento de um produto vão se concretizar, precisamos nos comprometer em meio à dúvida. Quando elas não se efetivam, mudanças acontecem e um novo ciclo recomeça com novos aprendizados. A incerteza é, portanto, um componente fundamental da inovação — se estamos criando algo novo, por regra não podemos ter certeza sobre os resultados.

Complexidade se apresenta também em várias circunstâncias. Partindo de um exemplo mais técnico, quando implementamos uma classe que possui muitos atributos e, portanto, muitos estados possíveis, é difícil contemplar todos esses estados de forma que o objeto se comporte apropriadamente em todos os casos em que ele é utilizado.

Boas práticas de desenvolvimento foram pensadas para lidar com essa complexidade ao longo do tempo, como ao implementar e manter uma suíte de testes automatizados, mas não nos livramos dela. Cada nova abstração também possui seus custos. De um ponto de vista mais abrangente, a complexidade também gera desafios quando precisamos avaliar resultados de mudanças. Dado que um produto está inserido em um contexto com inúmeros fatores, como saber se a modificação implementada realmente foi a causa da mudança de algum indicador?

Por fim, acredito que a ambiguidade deve ser o maior dos desafios. Por muitas vezes estaremos em situações onde não é possível racionalizar uma resposta certa e nos colocaremos em certos paradoxos. Devo focar no desenvolvimento de uma solução, ou ainda preciso explorar mais possibilidades? Devo tomar uma decisão baseada nos dados que eu tenho, ou usar minha intuição que contradiz os dados atuais? Devo melhorar esse produto, ou ele já está bom o suficiente e devemos priorizar outras frentes? O que seria bom o suficiente? Qual é a métrica certa?

Todas essas propriedades implicam que descobertas precisarão acontecer por meio de interações e que vão acontecer erros. Em resposta a isso, surgiram metodologias – aprofundaremos mais sobre elas adiante.

O que poucas pessoas falam é que todas essas características possuem grande potencial de gerar sofrimento. Sabemos que erros vão acontecer, e precisamos aprender com eles, mas eles não deixam de ser doloridos por causa disso. Mudanças bruscas e volatilidade vão sempre gerar estresse. Insegurança sobre o futuro e dificuldade de preparar-se para ele causarão ansiedade. Complexidade e ambiguidade levarão a falhas e, possivelmente, à perda de confiança entre as relações.

Nesse ambiente, decisões erradas acontecerão no âmbito individual e coletivo, mas todos pagarão com sofrimento. Entretanto, o estilo de gestão determinará muito sobre a intensidade desse sofrimento, embora não o resolva por completo.

Resumo

- Programar é diferente de desenvolver software. Quando programamos, o software é o objetivo em si, e quando desenvolvemos software, ele é um meio para alcançar um objetivo que está fora dele. Essa diferença possui implicações enormes. Compreender essa diferença é fundamental para adotar uma abordagem sistêmica de gerenciamento de projetos, de forma a reduzir sofrimento.
- Desenvolver software se encaixa na categoria do trabalho do conhecimento. Lidamos com um ambiente volátil, incerto, complexo e ambíguo. Nesse ambiente, o desempenho é medido pela qualidade de nossas escolhas e não pela quantidade de trabalho produzido.

Referências

[1] STERN; Stefan. Fond farewell to a brilliant thinker. *Financial Times*, 2009. Disponível em: https://www.ft.com/content/0168c7de-cd7e-11de-8162-00144feabdc0. Acesso em 16 set. 2022. Tradução livre do autor.

[2] WARTZMAN; Rick. What Peter Drucker knew about 2020. *Harvard Business Review*, 2014. Disponível em: https://hbr.org/2014/10/what-peter-drucker-knew-about-2020. Acesso em 16 set. 2022.

CAPÍTULO 2
Estilos de gestão e suas consequências

Nesta primeira parte, falaremos sobre a influência do estilo de gestão de projetos na sustentabilidade humana das organizações e na saúde mental dos indivíduos. Para exemplificar essa influência, contaremos uma história e apontaremos sete falhas comuns em equipes de desenvolvimento no decorrer da narrativa.

2.1 Uma startup lutando para sobreviver

Samuel, Daniel e Sueli trabalhavam juntos em uma startup em crescimento. A empresa chamava-se HighFive Delivery e propunha um modelo inovador de venda, compra e entrega de refeições online. Eles ainda eram pequenos, com algumas dezenas de funcionários, mas tinham construído um produto realmente bom e o negócio dobrara de tamanho nos últimos dois anos consecutivos.

Apesar do crescimento acelerado, havia um desconforto imenso na diretoria. Todos estavam preocupados com a presença de um concorrente capitalizado que tinha surgido no último ano, com um modelo de negócios semelhante. Com muito dinheiro de venture capital, *o concorrente investia de forma agressiva em marketing e abocanhava uma boa fatia do mercado.*

Naquela situação, a HighFive Delivery corria contra o tempo para se tornar mais competitiva e não ser engolida pela concorrente. Para isso, trabalhavam em várias frentes do produto ao mesmo tempo: na reformulação da experiência no aplicativo para aumentar a retenção; na criação de programas de indicações para aumentar a base de usuários; e na incorporação de funcionalidades de rastreio na entrega, porque a empresa concorrente já oferecia isso.

Erro 1 — Guiar-se pelo medo da concorrência

Já no começo da história, percebemos uma organização que atira para todos os lados. Parece não haver uma estratégia clara sobre como a HighFive Delivery pretende crescer. Há várias frentes diferentes de produto ao mesmo tempo e ainda não sabemos se as equipes são capazes de gerenciá-las de forma sustentável. Embora todas as frentes façam sentido para o negócio, a verdadeira motivação que está guiando as decisões é o medo da concorrência.

Há uma diferença entre jogar para ganhar e jogar para não perder. Quando o medo domina o ambiente e as decisões estão dirigidas para não perder, busca-se segurança. É em busca de segurança que dizemos "sim" a todas as possibilidades. Temos medo de apostar em uma ou poucas ideias, então queremos tudo.

Com muitos projetos em andamento, a organização como um todo fica mais lenta. Não importa se mais pessoas são contratadas e alocadas nos projetos. Em curto prazo, a capacidade de entrega será comprometida — e isso é péssimo para quem corre contra o tempo. As consequências dessa abordagem não param nos atrasos e na lentidão, porque há um resíduo em forma de estresse que recai sobre as pessoas. Os assuntos inacabados geram uma tensão que acaba reduzindo a capacidade das pessoas de tomar decisões e fazer avaliações.

As decisões orientadas por um viés positivo, em vez do medo, aceitam que há uma restrição de capacidade e que devemos apostar em poucas coisas e nos comprometer com elas. Com isso, as chances de construir um ambiente sustentável crescem consideravelmente, assim como a capacidade de entregar o que precisa ser feito.

Dentre as várias melhorias que a HighFive Delivery estava desenvolvendo, uma delas era a substituição do serviço de pagamento. O fornecedor atual, que era responsável por realizar as

cobranças, ficava instável em horários de pico e essa era a principal causa de reclamação dos usuários. Daniel, um dos diretores da empresa, chamou pessoalmente Sueli, uma das gerentes de produto, para lhe passar sua opinião. "Eu revisei os números que a equipe de suporte agregou. Precisamos resolver nossos problemas no pagamento. Eu sei que sua equipe está lidando com outras demandas, mas você acha que consegue encaixar a substituição do serviço que realiza a cobrança?". Na reunião quinzenal que Sueli tinha feito no dia anterior com sua equipe, um dos desenvolvedores estava com 90% de suas tarefas concluídas e estaria livre para iniciar outros projetos na semana seguinte. "Tudo bem, mas faremos um orçamento no início da semana que vem, para entender se isso cabe no nosso planejamento".

Erro 2 — Falta de clareza sobre o propósito do projeto

Quando o projeto de substituição do serviço de pagamento foi considerado, não houve aprofundamento sobre o propósito dele.

Daniel tinha a informação de que problemas com pagamento eram as reclamações mais recorrentes dos clientes. Com isso, era óbvio que esse problema teria que ser corrigido para vencer a concorrência — mas, naquele momento, ninguém esteve disposto a esclarecer as hipóteses e premissas por trás dessa proposta. Quais eram as reais consequências dos problemas no pagamento? Quantos usuários estavam sendo afetados? Quais eram as causas desses problemas? Em que medida isso impedia a empresa de crescer? Substituir o fornecedor atual resolveria qual parte desse problema?

Quando seguimos cegamente um plano, sem mapear as premissas e entender claramente os resultados esperados, isso pode ser confortável no início, mas as consequências sempre chegam. As pessoas envolvidas no desenvolvimento da solução não serão capazes de avaliar o que merece um esforço adicional e o que está suficientemente bom. Os resultados esperados não são alcançados e todas as partes precisarão se desdobrar para encontrar a coisa certa em um tempo menor, e sob condições piores.

Samuel era esse desenvolvedor. Ele era uma pessoa confiável. Estava trabalhando há 2 anos na empresa e já conhecia bastante do produto. Aprendia rápido e entregava muitos cards toda semana. Era um profissional excelente e uma pessoa gentil — alguém com quem todos adoravam poder contar no grupo.

Na semana seguinte, Samuel e Sueli sentaram juntos e destrincharam a demanda. Mapearam pontos de mudanças e ainda aproveitaram para fazer melhorias no fluxo de pagamento, alterando interfaces e adicionando push notifications*, avisando o usuário quando o pagamento fosse aprovado. Samuel nunca tinha lidado com essa parte antiga do produto, mas ele avaliara a documentação da API do novo fornecedor e parecia simples. Eles orçaram juntos os* story cards *enquanto escreviam no quadro Kanban, e acreditavam que precisariam de 2 meses para entregar o projeto — isso já com uma gordura para imprevistos.*

Com esse orçamento, Sueli avaliou que poderiam iniciar a implementação. Samuel entregara os cards *atribuídos a ele no projeto anterior, embora não tivessem sido liberados devido a pendências de outros desenvolvedores.*

Erro 3 — Percepção incorreta sobre o trabalho concluído

Quando o novo projeto foi atribuído a Samuel, ele e Sueli consideravam que seu trabalho estava concluído no seu projeto anterior. Ele era programador e tinha entregado os *cards* atribuídos a ele. Embora outras partes não estivessem concluídas, estas não eram de sua responsabilidade.

Entretanto, essa é uma percepção errada sobre conclusão de tarefas. Na verdade, uma tarefa só deve ser considerada concluída quando se encaixa em pelo menos uma das seguintes opções:

- A tarefa foi entregue e foi possível saber se ela atendeu ou não seu propósito de forma satisfatória para as partes interessadas;

- O projeto é intencionalmente abortado.

Portanto, quando o programador abre seu *pull request*, o ciclo ainda está longe de se fechar. Ele ainda não conseguiu o feedback das partes interessadas para saber se o que ele fez foi bom ou ruim. Tampouco o projeto foi abortado. Nesse caso, o trabalho ainda está em andamento.

Da mesma maneira, se o software está em produção, mas bugs impedem que ele seja usado, a tentativa não foi concluída. Se ele estiver funcionando, mas não houver maneiras de saber se a modificação foi boa ou ruim, o projeto também não foi finalizado.

A percepção errada sobre a conclusão das tarefas faz com que haja mais trabalho em andamento do que parece. Com isso, pessoas passam a ser interrompidas e acumulam assuntos inacabados. Essa carga cognitiva tem um custo — falaremos mais sobre ela em capítulos adiante.

Quando Samuel começou, ele preferiu inicialmente desenvolver todas as novas classes e abstrações que ele sabia que seriam necessárias para a nova integração. Tudo correu bem por algumas semanas. O desenvolvimento fluía e tanto Samuel quanto Sueli estavam confiantes e satisfeitos.

Mas chegara o momento de começar a analisar como as novas partes se encaixavam no todo. Samuel estava adiando essa fase do projeto, porque sabia que precisaria ler muito código antigo, com práticas de programação que ele não concordava, e que seria desgastante. Deixar isso para depois foi algo que aconteceu quase instintivamente, como uma maneira de adiar esse desconforto.

Quando ele começou a se debruçar sobre aquele código, uma avalanche de problemas surgiram. Apesar de ele ter encontrado código duplicado espalhado por vários pontos do sistema, como ele já esperava, isso era o de menos. Ele também descobriu inúmeros casos de uso que eles não foram capazes de mapear no início do

planejamento. Pedidos poderiam ser cancelados por vários motivos. Falhas no pagamento poderiam também ter várias causas, com feedbacks diferentes para o usuário. Confirmações com o serviço de pagamento também aconteciam de forma assíncrona em algumas situações e eles não tinham previsto o desenvolvimento desse webhook *para receber esses* callbacks._

Erro 4 — Adiamento dos pontos de maior incerteza

Samuel decidiu adiar o sofrimento de lidar com partes obscuras do projeto e preferiu ter o prazer imediato do progresso fácil e rápido. Isso é algo que frequentemente fazemos. Procrastinamos tarefas importantes nos mantendo ocupados com o que é confortável. Com isso, Samuel se colocava em uma situação difícil, pois não reconhecia riscos. As informações contidas nos pontos de incerteza revelaram uma avalanche de novas tarefas nas últimas semanas, quando já havia pouco tempo para se reorganizar.

Quando iniciamos um novo projeto, normalmente começamos mapeando as partes dele — as histórias de usuários e as etapas para implementá-las. Nesse momento, é primordial reconhecermos quais são as partes fundamentais do projeto e onde estão os maiores pontos de incerteza. Essas duas informações devem influenciar a prioridade das atividades. Histórias importantes e com alto grau de incerteza devem ser iniciadas antes, porque possuem grande potencial de mudar o rumo do projeto. Se uma mudança for necessária, é melhor que seja reconhecida o mais cedo possível.

Com as novas informações coletadas ao analisar o código existente, Samuel já tinha percebido que a gordura planejada na definição do prazo seria utilizada. Ele sentiu ansiedade. De repente, tudo que parecia tão simples ficou nebuloso. Mas ele ainda estava confiante e continuou com o plano, adicionando os novos pontos de ajuste na lista de tarefas.

Quando um mês havia se passado, Samuel foi chamado para uma das reuniões diárias da equipe de que fazia parte anteriormente. As alterações que ele havia implementado antes do projeto atual tinham entrado na última liberação havia algumas semanas, e problemas começaram a acontecer. A equipe já estava investigando os erros por alguns dias, mas era difícil fazer debug deles, porque não aconteciam sempre. Parecia que aconteciam de maneira aleatória. Apesar de Samuel estar preocupado com o projeto das mudanças no fluxo de pagamento, ele se sentiu responsabilizado pela situação e queria ajudar.

Naquele momento, ele preferiu não falar das preocupações com sua gerente. Apenas disse que poderia ajudar por meio período durante a próxima semana e trabalhar em par com outros desenvolvedores para tentar encontrar o problema. Ele percebia que Sueli estava se sentindo pressionada e visivelmente irritada e não queria trazer mais problemas. Além disso, uma performance review *estava próxima e Samuel não queria gerar uma má impressão logo antes da avaliação.*

Erro 5 — Ausência de segurança psicológica

Nesse ponto, Samuel começou a sofrer as consequências da falta de clareza sobre o trabalho concluído. Sua contribuição no projeto anterior ainda não havia terminado e agora ele era chamado para resolver problemas complexos que aconteciam no ambiente de produção. Essa condição não foi considerada quando ele e Sueli se comprometeram com o projeto novo.

Ele percebeu que estava em apuros, mas preferiu não falar com sua gerente. Aqui, há duas formas de interpretar esse acontecimento. Por um lado, podemos colocar o ônus sobre Samuel — ele deveria ter tido coragem de relevar a situação para que a equipe pudesse se reorganizar. Por outro, será que o ambiente da organização naquele momento favorecia a comunicação de notícias ruins? Com esse olhar, o que Samuel sentiu naquele momento foi ausência de segurança psicológica.

Segurança psicológica é uma característica do grupo, e indica quão confortável as pessoas se sentem em colocar-se em posições de vulnerabilidade e falar o que pensam. Com o clima de tensão na empresa, Samuel não se sentiu capaz de indicar suas falhas para sua gerente, mesmo sendo uma pessoa comunicativa e respeitada pelo grupo.

Samuel começou a chegar mais cedo e sair mais tarde. De manhã, avançava na implementação da integração com serviço de pagamento e, à tarde, trabalhava em par com desenvolvedores para encontrar o problema que estava acontecendo em produção, e foi assim por toda a semana.

Na sexta-feira, eles conseguiram encontrar o motivo dos erros, que aconteciam por uma questão de concorrência entre duas funcionalidades novas. Quando descobriram, ficou fácil. Implementaram alguns ajustes por algumas horas e o problema estava resolvido. Era uma ótima forma de terminar a semana. Porém, Samuel imediatamente pensou sobre seu projeto que estava atrasado. Ele não planejara fazer horas extras naquele final de semana, mas sua cabeça não conseguia se desligar do trabalho. Por vezes, pegava-se pensando nos problemas não resolvidos enquanto assistia a episódios de uma série no sábado à noite.

Na semana seguinte, Samuel começou a fazer mais horas extras. Era uma corrida contra o tempo, então ele baixou a barra nas suas exigências pessoais de qualidade. Ele conseguiu ser muito produtivo e, na quinta-feira, começou a fazer testes locais de ponta a ponta das alterações. Mais uma vez, apareceu uma sucessão de problemas. As APIs não se comportavam exatamente como ele esperava e respostas não documentadas eram retornadas da API para a aplicação. "Como é possível um programador ser tão incompetente a ponto de não manter uma documentação atualizada?", ele pensou.

A essa altura, Samuel estava arrasado. Ele já não conseguia tomar decisões racionais e apenas reagia aos problemas que aconteciam quando realizava seus testes. Na semana seguinte, faltando duas semanas para a entrega do projeto, Samuel resolveu pedir ajuda.

Ele estava exausto. Comunicou para Sueli que o projeto estava em risco e que não conseguia apresentar o que faltava ser implementado ou corrigido. "Por que não pediu ajuda antes?", Sueli perguntou, irritada.

Ela também estava esgotada devido à pressão que sofria da diretoria — ela pensava que todo aquele peso estava sendo segurado por ela, enquanto os desenvolvedores iam para casa às 18h. Samuel não sabia responder por que não avisou antes. Depois de tudo, ele se sentiu culpado por não ter sido capaz de pedir ajuda quando era necessário.

Sueli alocou um novo desenvolvedor para ajudar Samuel e eles mapearam o que faltava para concluir a demanda. Pediram mais duas semanas após o prazo combinado inicialmente, e começaram a trabalhar dia e noite. No início, as coisas demoraram para ganhar ritmo, porque o novo desenvolvedor precisava se familiarizar com o novo código e as melhorias incorporadas no processo, mas o trabalho começou a ganhar uma boa cadência.

Erro 6 — Resposta com força bruta

Dentro desse cenário caótico, Samuel tentou vencer os desafios com força bruta. Começou a trabalhar mais. Ele não considerou que poderia ter chamado Sueli para rever o escopo do projeto. Quem sabe melhorias previstas poderiam ter sido negociadas e deixadas para uma entrega futura, mesmo que parte do trabalho já realizado precisasse ser descartada na primeira entrega.

Por muito tempo Samuel se manteve otimista; apenas nas últimas semanas pediu ajuda. Nesse momento, Sueli também teve uma abordagem de resolução na força bruta ao adicionar um novo

desenvolvedor para auxiliar Samuel. Mais uma vez não houve discussão sobre o escopo do projeto. Eles só viam uma opção: entregar tudo, custe o que custar.

Há duas consequências decisivas na conduta de resolver um atraso com força bruta — com mais horas de trabalho. Primeiro, não consideramos que, quando trabalhamos até a exaustão, a qualidade de nossas decisões piora consideravelmente. Quando falamos sobre desenvolver software, onde nossa produtividade depende enormemente da qualidade de nossas escolhas, essa consequência é terrivelmente prejudicial para os resultados almejados.

Além disso, quando adicionamos horas designando mais pessoas para trabalhar em um projeto, muitas vezes esquecemos a nova carga de alinhamento que será necessária. Um projeto de um mês não pode ser dividido em duas pessoas para ser entregue em duas semanas, ou em trinta pessoas para ser entregue em um dia. Cada nova pessoa que participa da implementação requer mais tempo de alinhamento para garantir que todos trabalhem de forma coordenada em direção a uma solução. Em curto prazo, o trabalho ficará inevitavelmente mais lento.

É evidente que mais pessoas poderão, sim, ajudar a entregar um projeto atrasado em menos tempo. Mas essa ajuda deve vir com o objetivo de melhorar a qualidade das escolhas, como ao questionar a arquitetura de uma solução ou contribuir com mais conhecimento para sua implementação — e não apenas em ter mais "braços" para entregar tarefas pendentes, embora isso também seja importante.

Quando faltavam 2 dias para o fim do prazo, eles finalmente abriram o pull request. *Nesse momento, eles estavam trabalhando em branches abertas há mais de dois meses, com milhares de linhas alteradas, e vários conflitos apareceram. Samuel, que achava que finalmente poderia respirar e trabalhar em outras coisas, não conseguia acreditar que precisaria de mais dias de resolução de*

conflitos e novos testes, já que eles não tinham implementado testes automatizados.

Nesse dia, Daniel recebeu novamente os dados agregados das principais reclamações de clientes e constatou que ainda estavam lá as reclamações sobre eventuais falhas no pagamento. Ele ficou furioso. Agora que a empresa precisava acelerar e sua equipe de desenvolvimento não conseguia cumprir um prazo sequer, chamou Sueli para saber o que estava acontecendo.

Sueli tentou explicar os desafios pelos quais seus times passavam, mas os ânimos se exaltaram. Daniel não se comunicava como se realmente quisesse entender o que estava acontecendo. No fundo, ele pensava que estava sendo enrolado, porque não confiava mais na sua equipe. Sueli, naquela situação, começou a agir com cinismo.

Já havia algum tempo ela não se via mais na empresa. Ela não percebia mais um crescimento pessoal, nem uma conexão de propósito por estar trabalhando ali. Depois dessa conversa, começou a aplicar para algumas vagas em outras empresas — o mercado estava aquecido, ela não precisava passar por aquilo.

Quando o merge *do* pull request *finalmente aconteceu, a sensação não foi de vitória. Novos projetos já estavam na fila para serem iniciados, e já estavam atrasados, e o ciclo recomeçaria. Agora, com a confiança das relações destruída.*

Erro 7 — Iterações longas

Quando Sueli e Samuel delimitaram o tamanho do projeto, eles não planejaram entregas graduais. Para Samuel, era tudo ou nada. Parte dos requisitos emergiu durante o desenvolvimento das tarefas, como de costume — e, por não haver iterações curtas, acumularam demandas.

Sem um plano para entregas parciais, a liberação do projeto se tornou um problema. Muitas alterações seriam liberadas ao mesmo tempo, enquanto o estado do projeto tinha mudado bastante no decorrer do desenvolvimento.

Percebo que, na maioria das vezes, há um consenso entre as pessoas que trabalham com software de que entregar o produto de forma gradual é mais saudável do que um modelo em cascata. Entretanto, as pessoas nem sempre entendem o propósito disso.

A proposta de entregar software constantemente não existe para incentivar que nós trabalhemos da forma mais rápida que for possível. Nós queremos entregar software de maneira incremental para gerenciar o caos proveniente da criação de um produto por meio de ciclos de feedback.

Esses ciclos acontecem em várias camadas do trabalho. Quando a pessoa desenvolvedora altera uma linha de código e sua IDE lhe aponta um erro, uma iteração aconteceu — ela fez uma modificação e recebeu um feedback de que cometeu um erro. Da mesma forma, quando uma função é modificada e um teste é executado, um feedback acontece e indica se a modificação atendeu seu propósito ou não. Testes automatizados e outras ferramentas contribuem para tornar esse ciclo mais rápido e entregar uma experiência de desenvolvimento mais agradável e fluida.

Portanto, podemos olhar essas iterações como uma maneira de reestabelecer ordem em um processo de criação que fundamentalmente gera caos. Quando fazemos modificações, inserimos caos no projeto, e quando recebemos um feedback positivo, estabelecemos ordem.

Fora do âmbito técnico, essa mesma dinâmica se aplica no domínio de negócio de maneira ainda mais crítica. Quando uma empresa lança um novo produto, ou uma nova funcionalidade, ela não pode ter absoluta certeza de quais serão os resultados daquela liberação. Por isso, na medida do possível, ela deve coletar feedbacks

enquanto desenvolve seus produtos, para poder ter indícios de que caminha na direção correta.

Compreender essa dinâmica é crucial para a sustentabilidade do sistema de trabalho. A dinâmica de caos e ordem, quando não gerenciada de maneira adequada, acumula uma grande quantidade de estresse e ansiedade nas pessoas. Aprofundaremos mais sobre esse assunto em capítulos seguintes.

Depois de tudo, pelo menos, as reclamações dos clientes finalmente foram sanadas. Novos problemas apareceram por um tempo, mas foram solucionados aos poucos.

Mas ninguém quis saber se isso realmente mexeu nos indicadores-chave da empresa. Na verdade, apesar de as falhas no pagamento da integração anterior serem frequentes, as ocorrências representavam uma fatia baixíssima das transações. Ainda assim, boa parte delas era simplesmente por falta de saldo no cartão de crédito do cliente.

Ninguém quis saber, o cinismo já tomava conta do ambiente.

Dali para a frente, Daniel continuaria a empurrar hipóteses não validadas para seus gerentes de produto, porque pensava que eles não tinham a visão necessária para melhorar os números da empresa. Para ele, sua equipe não entendia a urgência das demandas e apenas ele estava com a pele em jogo.

Sueli não questionaria, porque buscava agora fazer o mínimo necessário, para evitar conflitos com Daniel, e ir embora no final do dia. Ela não se sentia confortável para falar com ele, nem tinha energia para isso. Tendo que lidar com vários projetos ao mesmo tempo, ela preferia poupar o esforço da discussão. Como gerente de produto, o que ela menos fazia era pensar estrategicamente sobre o produto, e não percebia seu trabalho como um ambiente de crescimento pessoal. Sentia-se apenas como entregadora de cards, e não era isso que ela queria.

Samuel passou a orçar tarefas de forma extremamente conservadora e a evitar os projetos mais desafiadores. Ele buscava o que era mais confortável para ele, e não o trabalho que realmente precisava ser feito, porque perdera sua autoestima e confiança como desenvolvedor.

As consequências e o que não vemos

Nessa história, como consequência da conduta adotada por todos, houve uma quebra de confiança entre as relações.

Um desânimo geral tomou conta e ninguém quer estar em um ambiente assim. Depois de tudo, a entrega até resolveu as reclamações, mas não trouxe o resultado esperado. Quem sabe teria sido melhor não ter feito nada por 4 meses, em vez de ter implementado um projeto que destruiu as relações e não trouxe quase nenhum benefício.

Com frequência, histórias parecidas acontecem no mundo real, mas gerentes não conectam as consequências às práticas de gestão adotadas. Eles não conseguem ver que algumas decisões podem parecer boas no curtíssimo prazo, mas terríveis no longo prazo, pelo simples fato de que são pessoas que realizam o trabalho. Forçar acordos irreais pode fazer um projeto ser entregue em um prazo espetacular, mas, quando o custo é a confiança das relações e a saúde das pessoas, a empresa se colocará em uma situação difícil.

2.2 As pessoas lutando para sobreviver em uma startup

A história de Daniel, Sueli e Samuel na HighFive Delivery é fictícia, mas, caso você tenha alguns anos de experiência, é provável que tenha vivenciado derivações dela.

O desajuste entre a abordagem de gestão e o trabalho de desenvolvimento de software causam consequências que vão além do fracasso pontual do projeto. As pessoas podem ser profundamente afetadas, assim como a empresa.

As consequências sobre as pessoas vão variar. Uma pessoa inserida em um ambiente de pressão, com incentivos contraditórios, injustiças e relacionamentos tóxicos, pode até suportar esses aspectos com tranquilidade, enquanto outra pessoa pode sofrer efeitos severos na saúde. Apesar disso, é fácil perceber se a tendência que o contexto promove é positiva ou negativa.

É muito comum que empresas busquem compensações como resposta para seu ambiente de trabalho com elevado nível de estresse. Salas de descompressão e técnicas de relaxamento são importantes, mas elas são compensações. Para começar, por que as pessoas estão precisando se descomprimir dentro da empresa?

Nessa reflexão, é importante usarmos a gestão como ponto de partida, pois, quando falamos de saúde mental no trabalho, precisamos pensar em um sistema sustentável em sua totalidade. Falhas vão sempre acontecer, e haverá sofrimento e frustração no processo de desenvolvimento, mas a organização deve estabelecer um esquema de trabalho que se regenere ao longo do tempo. Para isso, uma abordagem de gestão que atenda às necessidades humanas é fundamental.

Quando isso não acontece, as pessoas procuram alternativas.

O pesadelo da rotatividade

Os efeitos de abordagens ineficazes de gestão, assim como outros aspectos que veremos adiante, podem elevar o índice de rotatividade da organização. Quando uma pessoa sai do time de desenvolvimento, é uma tragédia para a empresa, porque há um enorme custo associado a isso — fazer um processo seletivo de

qualidade é caro e a perda de produtividade em curto prazo é inevitável.

Essa tragédia é minimizada quando há um ambiente onde a gestão do conhecimento é bem administrada. Software bem organizado, documentações em dia e ciclo de desenvolvimento bem estabelecido com procedimentos automatizados reduzem o estresse causado pela saída de um desenvolvedor da equipe — mas há sempre uma perda gigantesca com o conhecimento que se vai.

> *"Homens e meses são intercambiáveis apenas quando uma tarefa pode ser dividida entre muitos trabalhadores que não se comunicam entre si. Isso é verdade quando se debulha trigo ou colhe algodão, mas não é sequer aproximadamente real quando se trata de programação de sistemas."* – Frederick P. Brooks Jr. [1]

Como Frederick P. Brooks Jr. observou em seu livro *O Mítico Homem-Mês*, escrito em 1975, nós não podemos reduzir o tempo de implementação de um projeto de forma linear ao acrescentar pessoas na equipe. Um projeto de 10 meses não pode ser feito em 1 mês por 10 pessoas. Essa lógica pode funcionar para categorias de trabalho em que não há interdependência entre os trabalhadores, mas não funciona no desenvolvimento de software.

Cada nova pessoa trabalhando em um time implica em uma nova demanda por alinhamento. Isso significa que há mais uma pessoa que precisamos garantir que recebe e envia informações precisas sobre o projeto em andamento. Resolver a sobrecarga de alinhamento e lidar com a interdependência é o que, no fim das contas, as metodologias de desenvolvimento propõem. Elas sugerem maneiras de resolvê-las de forma que a vida não se torne um inferno.

Portanto, nesse cenário, quando um desenvolvedor sai da empresa, a substituição nunca é simples. Mesmo que os profissionais tenham currículos semelhantes, o dinheiro não poderá comprar o

conhecimento das nuances do negócio que o desenvolvedor antigo tinha. Se uma desenvolvedora com 3 anos de experiência na HighFive Delivery sair, a empresa não encontrará no mercado um outro desenvolvedor com 3 anos de experiência na HighFive Delivery.

Quando uma empresa de software, que possui valor por produzir conhecimento, tem alta rotatividade de funcionários, ela patina ao fazer progresso, pois está a todo momento perdendo parte dessa construção de conhecimento.

A importância de poder se expressar

Para superarmos esses problemas e construirmos um sistema de trabalho sustentável para as pessoas, precisamos abordar projetos orientados pelo pensamento sistêmico. Nesse sentido, paramos de olhar as pessoas como engrenagens de uma máquina e interpretamos a empresa como um organismo, que se transforma e evolui.

A transição entre a visão analítica para a sistêmica ao gerenciar projetos de software se dá no surgimento de metodologias que incorporam ciclos de feedback. Já que o conhecimento é construído durante o desenvolvimento da solução, devemos acelerar esse aprendizado para sermos mais eficientes e eficazes ao entregar software, e fazemos isso quando tornamos o ciclo de feedback mais rápido — isso acontece no âmbito da pessoa desenvolvedora, ao testar seu próprio código, e, no da organização, ao validar suas hipóteses de negócio.

Esses ciclos de feedback podem acontecer de várias formas, como ao observar uma métrica de produto depois da liberação de uma *release*, ou ao rodar uma suíte de testes automatizados depois de uma mudança no código. Mas, em muitas ocasiões, esses feedbacks acontecerão por meio de iterações entre pessoas. Nesses casos, há uma premissa fundamental para que o ritual faça sentido: as pessoas precisam falar o que pensam.

Se as pessoas estão em um ambiente tóxico, onde elas não se sentem capazes de falar o que precisa ser dito, o feedback não existe e a interação deixa de fazer sentido. Não importa se o escritório não tem paredes, ou se a equipe faz reuniões diariamente: se a atmosfera do ambiente não permite que as pessoas falem o que pensam, a interação não atenderá seu propósito.

Em um contexto hostil, as pessoas buscarão se defender da agressividade com cinismo. O indivíduo para de se importar com os objetivos do grupo para se afastar daquela condição de sofrimento. Eles buscam fazer o mínimo necessário para não se incomodarem. Mas alguém é realmente feliz fazendo um trabalho pior do que é capaz?

Para que as pessoas possam falar, fazer perguntas e trazer notícias ruins quando necessário, devemos pensar em como as relações dentro de um time podem ser preservadas e fortalecidas, e este será o tema do próximo capítulo.

Resumo

- Decisões de gestão possuem impacto profundo na sustentabilidade humana das equipes. Erros comuns de gestão são: falta de controle sobre o trabalho em progresso; falta de clareza sobre o propósito do projeto; percepção incorreta sobre o trabalho concluído; adiamento dos pontos de maior incerteza; ausência de segurança psicológica; resposta com força bruta e iterações longas. Esses erros têm potencial de quebrar a confiança das relações e criar um ambiente tóxico para as pessoas;
- Cada pessoa interage com o ambiente de uma forma diferente, mas podemos perceber se a equipe possui uma tendência positiva ou negativa, em termos de qualidade das relações. Ambientes tóxicos promovem relações com menor qualidade e reduzem a eficiência das organizações;
- Podemos buscar um sistema de trabalho sustentável ao adaptar metodologias que incorporam o pensamento sistêmico por meio de ciclos de feedback.

Referências

[1] BROOKS, Frederick. P., Jr. *O mítico homem-mês*: ensaios sobre engenharia de software. Rio de Janeiro: Alta Books, 2018. p. 16.

Capítulo 3
Segurança para assumir riscos e estar vulnerável

Como falamos anteriormente, a atividade de desenvolver software possui uma forte interdependência dentro das equipes.

As pessoas desenvolvedoras não trabalham sozinhas, nem apenas com outras desenvolvedoras. Elas estão sempre inseridas em equipes multidisciplinares, seja direta ou indiretamente, e colaboram para poder entregar um software que resolva um problema que está fora dele.

Com isso, desenvolvedores precisam conversar entre si e entrar em acordo sobre como a solução para um determinado problema será implementada. Por exemplo, como será o design da API? Como os componentes de interface serão organizados? Quais serão os padrões de nomenclatura e separação de módulos? Como será a interface? Quais serão as tecnologias utilizadas? Como será o *pipeline* de liberação?

Ao mesmo tempo, as interações acontecem também com outros domínios da organização. Por exemplo, com base no orçamento alocado, quanto tempo temos para implementar o produto? Quais são as funcionalidades fundamentais que ele precisa ter? Qual plataforma precisa rodar a solução? Quão sofisticada precisa ser a experiência do usuário?

Essa interdependência implica que, para uma organização fazer progresso, ela precisa se comunicar bem e, quanto maior ela for, melhor ela tem que se comunicar para manter eficiência. No fim, muito do que aprendemos com práticas ágeis são melhores maneiras de se comunicar para permitir que equipes que lidam com conhecimento movam-se na mesma direção e possam resolver problemas de forma eficaz.

Mas há uma premissa fundamental sobre a qual poucas pessoas falam. Para que essas metodologias funcionem, as pessoas têm de estar dispostas a se colocarem em posições de vulnerabilidade. Fazer perguntas, falar sobre falhas e propor ideias são exemplos dessas situações.

Ninguém quer realmente parecer tolo na frente dos outros. Buscar a aceitação do grupo é algo que está na essência do ser humano. Inclusive, preferimos muitas vezes ficar em silêncio e manter nossa aparência ao invés de nos arriscarmos a falar algo e não ter a aprovação das pessoas. Essa é a assimetria entre risco e benefício ao falar.

Ficar em silêncio nesses momentos significa que escolhemos preservar nossa segurança pessoal em vez de realmente trabalhar pelo objetivo da organização. Calculamos que nosso risco interpessoal é mais relevante do que o risco de não ter um problema resolvido.

No entanto, equipes bem-sucedidas se comportam diferente. Nelas, as pessoas se sentem mais confortáveis em assumir riscos e responsabilidades. Elas participam, discutem e, no fim, nem sempre chegam a um consenso, mas conseguem comprometimento da equipe sobre as decisões.

A diferença dessas equipes é que elas têm segurança psicológica. O medo e o risco interpessoal nesses casos não são maiores que o senso de responsabilidade sobre o trabalho que precisa ser feito e, por causa disso, elas se sentem confortáveis para aprender, comunicar e falar sobre erros.

Segurança psicológica é, portanto, um fator determinante para satisfação e engajamento. Trabalhar em um ambiente em que temos liberdade para falarmos e sermos ouvido é mais satisfatório do que em um ambiente de medo e ameaças [1]. Por isso, criar a consciência de que, mesmo que você esteja se sentindo bem em colocar opiniões, outras pessoas podem não estar se sentindo assim, é

indispensável para estabelecer processos efetivos de promoção de segurança psicológica dentro de um grupo.

3.1 Quão nocivo é o medo?

Segurança psicológica é um tema com extensas publicações pela pesquisadora Amy C. Edmondson. Como ela relata em seus trabalhos, esse fator pode literalmente determinar a vida ou morte de pessoas [2].

No seu livro *The Fearless Organization*, ela descreve como o trágico acidente entre duas aeronaves Boeing 747 ocorrido em 1977 no Tenerife, ainda na pista de decolagem, poderia ter sido evitado se os dois copilotos presentes na cabine de comando tivessem conseguido se expressar de maneira mais efetiva. Naquele momento, mesmo com suas próprias vidas em risco, em pelo menos duas ocasiões eles manifestaram dúvidas sobre a atitude do capitão, mas não conseguiram se sobrepor à autoridade da cabine. A colisão ocorrida, por falta de comunicação clara, ocasionou a morte de 583 pessoas.

Nos projetos de software, o medo pode esconder outros tipos de riscos, como atrasos, bugs e problemas de segurança. Uma desenvolvedora pode ser cobrada por não ter comunicado uma notícia ruim em uma reunião antes de algo mais grave acontecer; mas, se queremos realmente evitar situações como essa, não devemos colocar todo o ônus sobre o indivíduo, porque segurança psicológica é uma característica do grupo. Devemos questionar se o ambiente favorece que as boas e más notícias sejam comunicadas abertamente e questionar como a cultura da empresa recebe falhas.

Alguns comportamentos podem estar bem colocados do ponto de vista moral, como no manifesto da cultura da empresa, mas ser inviáveis do ponto de vista prático, em reuniões com gerentes.

Gestores que não aceitam as incertezas inerentes do desenvolvimento de produtos, e que ainda não entenderam a dinâmica do trabalho do conhecimento, acreditam que incentivar equipes através do medo ainda é uma boa ideia. Isso pode funcionar em atividades em que os trabalhadores precisam apenas repetir tarefas e não interagem entre si, embora ainda existam formas melhores e mais humanas de incentivar pessoas.

Entretanto, como já vimos, esse cenário é completamente diferente quando desenvolvemos software. Estamos a todo momento fazendo acordos, interagindo e tomando decisões que vão impactar o trabalho de todos. A presença do medo colocará um empecilho nesses momentos de interação e tornará a comunicação menos eficaz.

3.2 Segurança do grupo e segurança individual

É importante deixar claro que criar incentivos através do medo é diferente do que comunicar o que está em jogo. No primeiro caso, você coloca o indivíduo na defensiva — ele buscará se defender do ambiente hostil, independente dos objetivos compartilhados. No segundo, caso haja segurança psicológica, você promove a união do grupo para superar as ameaças externas. Essas ameaças são o que estão em jogo e dependem do negócio da organização. Poderiam ser, por exemplo: a saúde de um paciente; um prejuízo financeiro para a organização ou para seus clientes; ou a sobrevivência da empresa. Nesses casos, a ameaça catalizará a energia das pessoas em direção à resolução do problema.

Para ilustrar essa diferença, temos a seguir uma matriz que foi retirada do livro *Great Boss Dead Boss*, de Ray Immelman [3], com algumas alterações. No livro, o autor traz metáforas sobre o comportamento tribal para apresentar *insights* sobre gestão organizacional.

	Mais segurança do grupo	Menos segurança do grupo
Mais segurança individual	(3) Complacência; pouco senso de urgência; sem necessidade de assumir riscos para inovar.	(4) Pessoas se juntam para resolver o que está em jogo e reforçar a segurança do grupo; cooperação.
Menos segurança individual	(2) Pessoas se sentem ameaçadas dentro do grupo; os indivíduos agem para atacar a tribo e se defenderem.	(1) Todos saem assim que têm oportunidade; alta rotatividade.

No quadrante (1), temos o pior cenário de todos. As pessoas não têm segurança dentro do grupo e, ao mesmo tempo, o grupo está em risco por motivos externos. Nesse caso, todos buscam sair assim que têm oportunidade.

No quadrante (2), temos uma situação comum em empresas bem consolidadas, mas que mantêm relacionamentos tóxicos. Não há uma grande ameaça externa que coloque o grupo em risco, mas, ao mesmo tempo, não podem confiar nos seus companheiros de trabalho. Um bom exemplo seria uma grande corporação na qual os funcionários estão atacando uns aos outros, lutando por suas promoções.

No quadrante (3), temos uma circunstância peculiar. Há segurança individual e segurança do grupo ao mesmo tempo. À primeira vista, tudo parece correr bem nesse caso. Porém, essa condição resulta em complacência. A falta de ameaças externas manifesta uma falta de propósito. Parece uma situação confortável, mas há aspectos negativos tanto para os indivíduos como para as empresas.

Para as empresas, esse estado leva a uma falta de urgência e necessidade de inovar. Quando as empresas se encontram nesse cenário, é porque, na verdade, ainda não reconhecem os riscos ou não conseguem articulá-los para seus colaboradores.

Para os indivíduos, esse contexto também é negativo. Como já falamos em capítulos anteriores, ninguém está plenamente satisfeito em superar desafios menores do que se sente capaz. Nessa posição, os indivíduos se sentirão estagnados na carreira e começarão a buscar alternativas.

Por último, o quadrante (4) é a combinação de fatores que levam à alta performance. Há um propósito claro para o grupo e as pessoas se sentem à vontade e confiantes para assumir os riscos necessários para alcançá-lo.

Sentindo-se parte

Precisamos reconhecer que, dada a estrutura social e desigualdade do mundo de que fazemos parte, a dinâmica do risco social é mais dura para algumas pessoas do que para outras.

Para uma pessoa que não se identifica dentro do grupo ou com suas lideranças, seu cálculo de risco social será mais desafiador — ela sentirá mais receio para falar e contribuir do que outras pessoas. Organizações têm muito mais a ganhar ao falar abertamente sobre esse assunto, do que ao fingir que o problema não existe.

O senso de pertencimento é crucial para criarmos um ambiente com segurança psicológica. Nesse sentido, a diversidade em uma organização tem um papel claro para contribuir nessa direção.

> *"Diversidade é ter um assento à mesa, inclusão é poder falar, e pertencimento é ser ouvido."* – Liz Fosslien e Mollie West Duffy [4]

3.3 Estratégias para construção de segurança psicológica

Muito bem, sabemos que segurança psicológica é importante, mas o que podemos fazer para construí-la?

Os próximos tópicos são baseados na proposta de Amy Edmondson, em seu livro *The Fearless Organization*, no qual ela apresenta um guia para lideranças. Acredito que as dicas são especialmente importantes para líderes, mas ainda muito úteis para todos da organização, já que a comunicação acontece bastante entre os pares.

A estratégia consiste em três passos:

- Preparar o ambiente;
- Convidar à participação;
- Responder de forma produtiva.

Preparar o ambiente

O primeiro passo é preparar um ambiente onde falhas são vistas como algo natural — não no sentido de que são boas, ou de que são sempre inevitáveis, mas de que a natureza do trabalho de inovação pressupõe que elas vão acontecer. Considerar que falhas não são um ponto fora da curva no desenvolvimento de um produto significa aceitar que o sistema de trabalho deve estar em constante processo evolutivo.

Empresas que incorporam o pensamento sistêmico por meio de metodologias ágeis possuem essa premissa implícita. Elas sabem que a evolução acontecerá por meio de iterações e feedbacks para que incertezas sejam superadas e, por isso, criam um clima de descobertas.

Entretanto, a compreensão sobre o trabalho realizada por meio de ciclos de feedback não surge em todas as áreas da organização ao

mesmo tempo. É possível que um time de desenvolvimento esteja comprometido em ter uma postura ágil nas suas decisões, mas sua diretoria, ou clientes, os puxe para um esquema em cascata. Nesse caso, quando você como pessoa desenvolvedora estiver sendo pressionada a aceitar acordos irrealistas, seu papel é apenas tentar comunicar os riscos e premissas da forma mais clara possível.

Dado que a segurança psicológica é uma característica do grupo, é possível que em uma mesma empresa haja equipes que trabalhem em um ambiente que favorece a colaboração, e outras que não. Minha sugestão é que as pessoas comecem pela busca em transformar seu próprio núcleo de trabalho. Nesse caso, mesmo que exista um ambiente hostil ao redor, podemos começar a transformar o clima social ao, lentamente, aumentar os vínculos de confiança.

Outro aspecto importante para preparação do ambiente é reforçar o propósito do trabalho — como falamos antes, é comunicar o que está em jogo para o grupo. Equipes ágeis colocam isso em prática por meio de, por exemplo, *story cards*. A todo momento está sendo comunicado entre os membros da equipe a hipótese de proposta de valor que se busca por meio da implementação de uma parte do software. As consequências sobre o trabalho executado ou não executado ficam mais claras, e o convite à participação fica melhor porque existe relevância sobre o que está sendo feito.

Convidar à participação

O próximo passo é fazer um bom convite à participação das pessoas. Para isso, primeiro temos que aprender a fazer boas perguntas.

Você não deve conseguir levantar muitas informações sobre a qualidade do trabalho entregue por sua equipe se fizer uma pergunta como "Quais erros cometemos na última semana e quem são os responsáveis?". Claramente, essa frase tem uma conotação negativa, e as pessoas poderão pensar:

- O que acontecerá com os responsáveis?

- Será que haverá uma punição?
- O que meus colegas de trabalho pensarão de mim se eu falar algo?

Nesse exemplo, podemos refrasear para: "Tudo correu bem na última semana?". Essa pergunta é mais convidativa à participação. Ela comunica uma premissa de que desejamos que as coisas tenham corrido bem, embora seja possível que não tenham. Quando o objetivo é mapear riscos e melhorar o sistema de trabalho, ela é mais eficaz do que a pergunta anterior.

Nós subestimamos o valor de uma boa pergunta porque estamos mais preocupados com nossa habilidade de analisar e responder. Entretanto, no trabalho do conhecimento, apenas uma boa pergunta tem o potencial de fazer emergir a inovação que uma organização precisa.

O que são boas perguntas?

Segundo os autores do livro *The Art of Powerful Questions*, Eric Vogt, Juanita Brown e David Isaacs, uma boa pergunta possui as seguintes características [5]:

Boas perguntas

- Geram curiosidade no ouvinte;
- Estimulam conversas reflexivas;
- São instigantes;
- Fazem emergir premissas subjacentes;
- Convidam à criatividade e a novas possibilidades;
- Geram energia e movimento;
- Canalizam atenção e focalizam investigação;
- Permanecem nos participantes;
- Tocam significados profundos;
- Evocam mais perguntas.

Para eles, podemos aprender a formular questões assim ao compreendermos a arquitetura das perguntas em três dimensões:

- Construção;
- Escopo;
- Premissas.

Construção

Na dimensão da construção, devemos prestar atenção nas palavras utilizadas na frase. Uma mesma questão pode ser formulada de maneira que provoque uma resposta "sim" ou "não", ou de forma que instigue respostas mais profundas. Por exemplo:

- Você está satisfeito com seu crescimento profissional nos últimos três meses?
- Quando você esteve mais satisfeito com seu crescimento profissional nos últimos três meses?
- O que proporciona mais satisfação e crescimento profissional para você?
- Como você pensa que podemos alinhar os objetivos da organização com seus anseios de crescimento profissional?

Nos exemplos acima, as últimas perguntas são mais poderosas do que as primeiras. Elas despertam mais curiosidade e reflexão e têm maior potencial de descobertas.

Podemos imaginar essa situação em uma reunião de feedback. De um lado, a pessoa que recebe a pergunta pode reagir apenas respondendo: "sim, estou satisfeito". De outra maneira, reflexões mais profundas serão provocadas:

- Como eu me imagino profissionalmente no futuro?
- Quais são meus pontos fortes?
- Em quais projetos da empresa eu sinto que gostaria de me envolver mais?

Existem algumas diferenças entre as versões dessa pergunta, e a principal delas é a palavra usada para iniciá-la.

Mais poderosas
Por que
Como
O que
Quem, Quando, Onde
Qual, Sim ou Não
Menos poderosas

Geralmente, perguntas que começam com "por que" ou "como" são convites mais poderosos para a cooperação do que perguntas que começam com "quem" ou "quando". Isso não é uma regra, e também não significa que essas perguntas são mais relevantes para todas as situações. Entretanto, alternar esse componente da estrutura é um bom começo para começar a praticar a criação de questionamentos mais interessantes.

Escopo

A segunda dimensão é o escopo das perguntas. Veja os seguintes exemplos:

- Como podemos melhorar a qualidade *dessa classe*?
- Como podemos melhorar a qualidade do *módulo de geolocalização*?
- Como podemos melhorar a qualidade do *produto*?
- Como podemos melhorar a qualidade da *empresa*?

Podemos observar que o escopo das perguntas acima cresce gradativamente. Para fazermos um convite adequado à participação das pessoas, devemos construir nossos questionamentos de forma

alinhada ao escopo em que as pessoas podem contribuir e tornar isso explícito de acordo com as nossas necessidades.

Premissas

Por último, temos a dimensão das premissas. Toda pergunta carrega uma premissa que deve ser reconhecida. Por exemplo:

- Como podemos evitar a falta de memória em nossos servidores?
- Como podemos responder melhor quando faltar memória em nossos servidores?

Na primeira pergunta, há a premissa de que podemos definitivamente resolver o problema que causa falta de memória em servidores. Na segunda, a premissa é de que podemos criar mecanismos ou procedimentos para responder melhor à falta de memória. Essa diferença levará as pessoas a explorarem soluções distintas para um mesmo problema. Questionar a premissa implícita na mensagem e alterná-la durante a avaliação de um assunto é uma excelente técnica para fomentar curiosidade e criatividade.

Em outro exemplo, imagine um cenário em que houve um atraso na entrega de um projeto — ele era mais complexo do que se imaginava e isso teve consequências severas para a empresa. Poderíamos investigar as causas e soluções para esse problema de diversas maneiras:

- O que poderíamos ter feito diferente para que o atraso não acontecesse?
- Como poderíamos ter respondido melhor ao atraso para reduzir suas consequências?
- Por que não foi possível identificar o descompasso entre o tamanho do projeto e o prazo com mais antecedência?
- Por que houve consequências tão severas?

Novamente, o que muda nesses questionamentos são as premissas implícitas na pergunta, que levam a investigações distintas para lidar com um mesmo problema.

Responder de forma produtiva

Após um convite apropriado, com uma boa pergunta, devemos responder à participação das pessoas de forma positiva e produtiva.

Para isso, é preciso, primeiramente, expressar apreciação pela participação. Como já sabemos, nem sempre é fácil para as pessoas se colocarem na condição de risco social e falarem o que pensam, embora isso seja necessário. Reconhecer esse esforço e engajamento pelo que está em jogo é importante para encorajar que isso continue a ser feito.

Além disso, é importante ter em mente que o objetivo é estabelecer uma conversa autêntica. Isso significa verdadeiramente ouvir o que as pessoas têm a dizer — com atenção, curiosidade e sem fazer julgamentos prematuros. As perguntas devem ser honestas, não um mecanismo retórico para comunicar uma ideia já formulada.

> *"Muitas das questões mais provocativas que são vitais para o futuro de uma organização são primeiramente descobertas nas linhas de frente, no meio da ação do dia a dia."* – Eric Vogt, Juanita Brown e David Isaacs [6]

Devemos encorajar que diferentes perspectivas emerjam e transformar o conhecimento em ações concretas. Com isso, as pessoas perceberão que vale a pena contribuir.

Resumo

- Quando escolhemos falar, assumimos um risco social. Muitas vezes, as pessoas preferem ficar em silêncio em vez de colocar sua reputação com o grupo em risco. Quando isso acontece, o risco social foi maior do que o risco de não ter um problema da organização resolvido;
- Metodologias ágeis possuem a premissa de que as pessoas falam o que pensam — se isso não acontece, elas não funcionam como deveriam;
- Segurança psicológica, ou seja, quão seguras as pessoas se sentem ao se colocarem em posições de vulnerabilidade, é um componente fundamental para a formação de uma equipe engajada e eficiente;
- Podemos promover segurança psicológica ao preparar adequadamente o ambiente, fazer boas perguntas e responder de forma positiva e produtiva.

Referências

[1] EDMONDSON, Amy C. *The Fearless Organization*: Creating Psychological Safety in the Workplace for Learning, Innovation and Growth. John Wiley & Sons Inc., 2018. p. 41-42.

[2] *Ibidem*, p. 79-82.

[3] IMMELMAN, Ray. *Great Boss Dead Boss*: How to exact the very best performance from your company and not get crucified in the process. Phoenix, Arizona: Stewart Philip International LLC, 2003.

[4] FOSSLIEN, Liz; DUFFY, Mollie. *No Hard Feelings*: The Secret Power of Embracing Emotions at Work. New York: Portfolio (Penguin Publishing Group), 2019. p. 185.

[5] VOGT, Eric E.; BROWN, Juanita; ISAACS, David. *The Art of Powerful Questions*: Catalyzing Insight, Innovation, and Action. London: Whole Systems Associates, 2003. p. 4.

[6] *Ibidem*, p. 12.

Capítulo 4
Desenvolvimento centrado no ser humano

Até agora, falamos sobre como o trabalho da pessoa desenvolvedora não é apenas escrever código; como uma gestão inapropriada pode comprometer a sustentabilidade do sistema de trabalho; e por que a segurança psicológica das equipes é um fator primordial para elas terem boa performance e engajamento.

Neste capítulo, aprofundaremos algumas práticas de desenvolvimento sob a perspectiva das necessidades humanas que são atendidas por elas — e não apenas sobre como funcionam e os resultados que podemos obter com elas. Com esse olhar, as necessidades humanas atendidas são um meio para se obter os resultados de negócio que almejamos.

Reconhecer essas necessidades é o primeiro passo para notarmos o que está passando despercebido para, depois, sermos capazes de contemplá-las com intenção e qualidade.

4.1 Revisitando valores de Extreme Programming

A primeira referência que vamos rever é o *Extreme Programming* (XP).

Como Kent Beck relata em seu livro *Extreme Programming Explained: Embrace Change* e também em algumas palestras, o XP surgiu durante uma consultoria que ele realizou na Chrysler. A empresa passava por dificuldades para implementar um novo

sistema para gerenciar as folhas de pagamento, e Kent Beck encontrou uma equipe arrasada e exausta naquele contexto. Eram pessoas qualificadas, mas estavam em um projeto sem esperança.

Nesse cenário, ele propôs à empresa três opções: continuar como estava; terceirizar o desenvolvimento; ou recomeçar o projeto. Dadas as circunstâncias, continuar o projeto da mesma maneira não era uma boa opção. Terceirizar também não parecia uma boa ideia, porque eles realmente contavam com pessoas competentes, capazes de implementar o projeto e que entendiam o problema que precisava ser resolvido. Portanto, sobrou a última opção. Eles decidiram dar uma semana de folga para todos e recomeçar.

Naquele momento, com Kent Beck encarregado pelo projeto, o XP emergiu. Com iterações curtas, definição de histórias, programação em pares, testes unitários e tudo mais, ele começava a se materializar como uma metodologia.

> *"XP é minha tentativa de reconciliar humanidade e produtividade em minha própria prática de desenvolvimento de software e compartilhar essa reconciliação."* – Kent Beck [1]

Comunicação

No capítulo anterior, falamos bastante sobre a importância de uma comunicação eficaz para a maneira como uma equipe de software trabalha.

Com a condição de interdependência que há entre os desenvolvedores, para que uma equipe possa avançar rapidamente ela precisa ser eficaz na sua comunicação. Pessoas desenvolvedoras precisam escrever boas funções, de maneira clara e sucinta, para que outras pessoas no futuro entendam seu propósito e funcionamento rapidamente. Os projetos como um todo devem ser compostos por alguns padrões de organização, para que novos colaboradores possam ganhar autonomia no menor período de

tempo possível. Gerentes de produto devem escrever *story cards* diretos e concisos para que a solução implementada tenha um bom encaixe com o problema. Essa dinâmica se estende por todos os níveis da organização.

Junto dessas necessidades, há também o anseio intrínseco das pessoas de fazerem parte de algo maior. Nós nos sentimos bem quando nos identificamos dentro de um grupo onde podemos ouvir e ser ouvidos, e buscamos essa identificação.

Nesse caso, comunicação está no fundamento do que dá sentido a um grupo. É por que as pessoas se comunicam e colaboram por um objetivo em comum que uma equipe se diferencia de um conjunto de pessoas que trabalham de maneira independente.

Esse valor desmistifica a ideia de que pessoas desenvolvedoras trabalham sozinhas. Elas não apenas recebem informações para entregar tarefas, como também são responsáveis por colaborar e devolver conhecimento importante para o futuro da empresa.

Simplicidade

O valor da simplicidade aparece para desconstruir a ideia de que o design perfeito de uma solução precisa vir *a priori*. Buscamos primeiro a solução mais simples possível que poderia resolver de forma satisfatória um determinado problema, dadas as restrições impostas pelo contexto. Isso se contrapõe à alternativa de desenhar uma solução perfeita de antemão, orçar seu custo e dividir o trabalho.

Pressupondo que é impossível desenhar a solução perfeita *a priori*, a busca pela simplicidade direciona a equipe de maneira a escapar de um clássico problema de projetos de software: o acúmulo insustentável de hipóteses.

Quando seguimos o plano de desenvolvimento de uma solução robusta *a priori*, uma quantidade enorme de premissas e hipóteses é colocada em campo. Por acreditarmos que o problema abordado

está completamente compreendido, desenhamos um fluxo sofisticado para a experiência dos usuários. Por desenharmos um fluxo sofisticado para a experiência dos usuários, precisamos de soluções técnicas também sofisticadas. Por precisarmos de soluções técnicas sofisticadas para entregar a experiência, temos que pensar em maneiras também sofisticadas para garantir que a aplicação escale na dimensão de usuários simultâneos que acreditamos que teremos. Nessa dinâmica, projetos sugam uma quantidade enorme de recursos desnecessários.

Todas essas hipóteses, quando invalidadas, influenciarão drasticamente outras decisões do desenho da solução. O problema é que uma grande parte dessas hipóteses só será colocada à prova quando o produto interagir com o sistema externo a ele, ou seja, quando o software for utilizado para resolver aquilo que ele se propõe a resolver. Até lá, todo o desenvolvimento estará acumulando caos através de hipóteses não validadas.

Por outro lado, quando buscamos soluções simples, continuamos apostando em hipóteses, mas elas serão menores, e isso muda completamente o jogo. As negociações serão menos tensas porque os projetos serão mais fáceis de compreender e a entrega de valor será mais rápida. Desse modo, o valor da simplicidade reforça o da comunicação, porque é mais fácil comunicar sobre soluções simples do que complexas.

Para elaborar soluções simples, as premissas dos projetos devem ser reconhecidas e questionadas. Isso deve acontecer do ponto de vista de cada parte interessada na implementação da solução, inclusive as pessoas desenvolvedoras. Ao avaliarmos a complexidade de uma solução e buscar alternativas, as pessoas que estão com a mão no código são as que terão mais propriedade para julgar e, por causa disso, devem participar do desenho do projeto.

Evidentemente, o conceito de simplicidade é relativo e dependerá do contexto em que o projeto está inserido. A solução mínima para um software que lida com dados pessoais de saúde de um hospital deve

contemplar camadas de segurança maiores do que um portal de notícias — assim como uma funcionalidade simples em um portal de notícias com uma grande audiência deve ter requisitos de escalabilidade mais complexos do que um software de gestão para hospitais. Portanto, o contexto vai impor suas restrições, e devemos buscar simplicidade a partir delas.

Feedback

O feedback é o mecanismo que direciona a evolução do software. Primeiro, buscamos simplicidade, para depois entender como o software deve evoluir dos feedbacks que extraímos.

Em capítulos anteriores já esmiuçamos o papel do feedback no ciclo de desenvolvimento de software. De forma sucinta, é através de feedbacks que conferimos ordem ao processo caótico de desenvolvimento de software.

Quanto mais rápido e mais barato for esse ciclo de obtenção de feedback, mais eficazes somos como profissionais e empresas, porque estamos sempre ajustando a direção. Com isso, desenvolvemos software com mais qualidade, porque ele faz o que se propõe a fazer; gerenciamos equipes de forma mais assertiva, porque as pessoas trabalham no que é mais importante no momento; e atendemos às necessidades dos nossos clientes, porque o produto entrega o valor que eles esperam. O conhecimento para essas descobertas, para que possamos fazer a coisa certa, só pode vir através de ciclos de feedback em várias camadas do trabalho.

Com a premissa de que o direcionamento do desenvolvimento do produto será reajustado ao longo do tempo, devemos aceitar uma parte fundamental dessa dinâmica: **erros vão acontecer**.

Soluções simples não serão à prova de falhas na implementação. É possível que o feedback obtido após a liberação de um produto seja: ele contém muitos bugs e não funciona para resolver o que se propõe. Além disso, falhas podem acontecer no encaixe problema-

solução. O software pode não ter bugs e fazer exatamente o que se propõe, mas ele não resolve um problema real de maneira satisfatória.

Por fim, falhas podem acontecer inclusive na própria avaliação do feedback. Muitas vezes nós não teremos todas as informações necessárias para tomar uma decisão de ajuste de rota, e erros podem acontecer nesse momento. Mesmo com métricas muito claras, sempre haverá algum nível de ambiguidade na interpretação dos resultados, e teremos que fazer apostas sobre os caminhos a seguir.

Portanto, erros são parte fundamental da dinâmica de desenvolvimento de software. Aprender a reconhecê-los e agir sobre eles é crítico para o sistema de trabalho funcionar. Sabemos que temos que aprender com nossos erros, mas é mais fácil falar do que fazer. Aprender com erros envolve lidar com emoções fortes e isso será pauta de capítulos adiante.

Coragem

Os valores anteriores estão bastante conectados com a forma como trabalhamos. Quando Kent Beck propôs o XP, a metodologia representava uma oposição ao modelo em cascata altamente burocrático que muitas empresas adotavam.

Coragem, por outro lado, traz algo diferente. Ela fortalece os demais valores por uma virtude humana, e não exatamente por uma nova prática ou maneira de entender o trabalho.

> *"Se coragem, sozinha, é perigosa, combinada com os outros valores ela é poderosa. A coragem para falar verdades, agradáveis ou desagradáveis, promove comunicação e confiança. A coragem para descartar soluções que falharam e buscar novas encoraja simplicidade. A coragem para buscar respostas reais e concretas cria feedback."* – Kent Beck [2]

A presença da coragem como um dos valores do XP implica que pessoas desenvolvedoras precisarão superar medos. Esses medos podem vir de várias formas, como quando assumimos responsabilidade sobre um projeto, mesmo sabendo que não temos todo o conhecimento necessário para implementá-lo e teremos que aprender durante o caminho; quando precisamos comunicar notícias ruins e desapontar pessoas; ou quando precisamos fazer algo que nunca fizemos antes.

Como vimos no capítulo passado, a presença de segurança psicológica no ambiente pode determinar muito nesse sentido — impulsionando ou reduzindo a coragem das pessoas para tomarem decisões. Contudo, mesmo com um espaço favorável, superar esses medos sempre demandará coragem individual. Segurança psicológica não é capaz de remover completamente a dor.

Respeito

Por último, temos o valor do respeito. Todos os valores anteriores dependem de tratarmos as pessoas como seres humanos e respeitarmos suas contribuições.

> *"Se membros de um time não se importam uns com os outros e com o que estão fazendo, o XP não vai funcionar. Se os membros de um time não se importam com um projeto, nada pode salvá-lo."* – Kent Beck [3]

É fácil imaginar (ou lembrar de) um lugar em que as pessoas são agressivas nas suas interações e entender como a ausência de respeito impede que os valores e práticas do XP funcionem. Como seria possível ter uma comunicação eficaz quando não há respeito mútuo entre as pessoas? Sem uma comunicação eficaz, como seria possível colaborar para perseguir simplicidade nas soluções? Sem

simplicidade, como seria possível iterar e obter feedbacks verdadeiros? Por fim, sem relações de respeito, por que a coragem importaria?

Ao mesmo tempo em que é fácil perceber sua importância, é difícil definir o que é respeito. Apesar de uma comunicação violenta associar-se à falta de respeito, a ausência de violência na comunicação não necessariamente implicará em uma relação de respeito.

Algumas pessoas também associam respeito com obediência. Em um ambiente com hierarquia rígida, como em uma escola, os alunos demonstram respeito aos professores ao ficarem em silêncio enquanto eles falam.

Não tratar com violência e ouvir com atenção são características que fazem parte de uma relação respeitosa, mas sabemos que respeito é ainda algo além disso. Respeito tem a ver com tratar o outro com dignidade — com reconhecer e atribuir valor às suas contribuições.

Quando interrompemos o trabalho focado de uma pessoa, por exemplo, podemos considerar que estamos desrespeitando seu tempo e energia. Consideramos que nosso assunto é mais importante do que qualquer outra coisa que ela esteja fazendo, e impomos a nossa própria prioridade para a outra pessoa. Nós obtemos o que queremos, com os custos da interrupção, que ficam com o outro. Naturalmente, alguns assuntos valem uma interrupção, mas devemos fazer essa avaliação tratando os colaboradores com respeito, considerando que eles estão fazendo um trabalho tão importante quanto o nosso.

Dessa forma, respeito deve ser a base para estabelecer relações de qualidade e confiança.

Esses valores levaram ao surgimento de uma série de práticas, como programação em pares, implementação de testes automatizados, estabelecimento de um processo de integração contínua, criação de

story cards, entre outras. Apesar disso, conseguimos ver que seus valores estão centrados nas pessoas e não nas práticas em si.

A metodologia tem muito mais a ver com a forma como as pessoas interagem, e como o fluxo de trabalho acontece, do que como cada um deve trabalhar individualmente, porque são nesses domínios que podemos realizar mudanças realmente significativas para alavancar resultados de uma organização — embora práticas individuais também sejam importantes.

O *Extreme Programming* pressupõe que o design apropriado de um software, que resolve um problema de forma satisfatória, deve emergir durante sua implementação. Depois, devemos refatorar a solução em busca de uma abordagem mais sustentável de evolução do produto ao longo do tempo. Essa sustentabilidade se dá ao considerarmos que são pessoas em interação que desenvolvem o software, e não pessoas isoladas ou máquinas.

A metáfora de que eu mais gosto, que comunica essa diferença precisamente, é a do chef e os cozinheiros. Temos duas formas de traçar um paralelo na dinâmica da cozinha de um restaurante com o desenvolvimento de software. Por um lado, podemos considerar que os chefs são as gerentes de produto e designers. Elas determinam o que deve ser feito e são responsáveis por criar a receita dos pratos que serão vendidos. Estes seriam os requisitos do software no nosso contexto. Os programadores seriam os cozinheiros, replicando de forma mais precisa e rápida que for possível essas receitas. Além do design da solução, os chefs também são responsáveis por determinar como os pratos devem ser entregues, ou seja, como deve ser o processo de desenvolvimento do produto.

Por outro lado, metodologias como o *Extreme Programming* colocam todas as pessoas (gerentes de produto, designers e desenvolvedoras de software) como chefs — ou seja, todas são responsáveis por colaborar para definir o que deve ser feito e como deve ser feito. Os cozinheiros nessa metáfora são as próprias máquinas, replicando a solução implementada e colocando o software em execução.

Compreender essa diferença é crucial para que seja possível estabelecer um sistema de trabalho sustentável. As necessidades das equipes de desenvolvimento devem ser contempladas de um ponto de vista estratégico da empresa para que as soluções certas, que respeitam as restrições do negócio, possam emergir do processo de criação. As pessoas desenvolvedoras devem estar em condições de darem o seu melhor — falaremos mais sobre como isso é possível adiante.

4.2 Efeito Zeigarnik e a importância do controle sobre trabalho em progresso

Em 1927, a psicóloga Bluma Zeigarnik publicou um artigo intitulado "*On finished and unfinished tasks*" [4], no qual observou que somos mais capazes de nos lembrarmos de detalhes de tarefas não finalizadas do que de finalizadas.

Zeigarnik percebeu que garçons em restaurantes conseguiam se lembrar de pedidos complexos enquanto eles eram preparados e servidos, mas esqueciam deles logo quando eram entregues. Esse fenômeno ficou conhecido como "Efeito Zeigarnik".

Ela teorizou que, quando estamos com uma tarefa inacabada, nossa mente cria uma tensão que se alivia apenas quando a completamos. Quando estamos nesse estado, há uma certa carga cognitiva nos persuadindo a resolver essa tensão — nossa mente fica incomodada e inquieta com a tarefa incompleta.

Quando lidamos com tarefas que podemos começar e finalizar em alguns minutos, ou pelo menos em um mesmo dia, essa tensão é boa. Ela nos ajuda a entregar o que precisamos. Entretanto, isso é raramente a realidade quando se trata de entregar software. Normalmente projetos demoram vários dias para serem

implementados, e, por isso, nós buscamos aliviar essa tensão ao dividi-lo em pequenas partes.

Mesmo assim, quebrar o projeto em pequenas partes não é suficiente. Como já descrevemos antes, no trabalho de desenvolvimento de software há uma forte interdependência entre as pessoas. Essa interdependência se apresenta nos momentos em que ocorre uma "passagem de bastão". Um exemplo disso é quando uma desenvolvedora implementa uma alteração, mas deve esperar a revisão de outra pessoa antes de fazer *merge* e liberá-la.

Essas passagens de bastão, ou *handoffs*, são o que geram uma grande confusão sobre a percepção do trabalho concluído. Para a pessoa que abriu o *pull request*, ela finalizou a tarefa e está disponível para receber um novo *card*. Mas a tarefa realmente não foi finalizada, porque a equipe ainda não recebeu um feedback sobre se ela resolve de forma satisfatória o que se propõe a resolver — ela ainda não foi liberada no ambiente de produção.

Com a falta de percepção sobre a quantidade de trabalho em andamento, a equipe é incapaz de perseguir eficiência de fluxo. As pessoas se comportam de forma a manter-se ocupadas fazendo o que consideram que são contratadas para fazer — como implementar a maior quantidade de código que for possível — e não se organizam para, de fato, colocar código funcionando na mão dos seus usuários.

Mas, em algum momento, essas pendências deverão ser passadas a limpo. Um *pull request* voltará com solicitação de ajustes. Uma alteração será liberada em produção, e serão identificados bugs que precisarão ser resolvidos. Enquanto isso, novas alterações precisam continuar sendo implementadas porque outras pessoas já esperam algo novo que foi acordado.

Portanto, essa confusão na gestão do trabalho em andamento e *handoffs* colocará as pessoas em uma condição improdutiva — de muita troca de contexto e alta carga cognitiva proveniente do

estresse gerado pelo trabalho inacabado. Nessas condições, as pessoas são incapazes de entrar no *estado de fluxo* descrito por Mihaly Csikszentmihalyi, situação em que os indivíduos estão completamente engajados em uma atividade e esquecem do tempo — não apenas para produzir o melhor trabalho que conseguem, mas também para estarem satisfeitos com ele.

Estabelecendo um fluxo de trabalho sustentável com Kanban

Para endereçar esses problemas, o Kanban como metodologia propõe duas práticas fundamentais. Primeiro, as pessoas devem ser capazes de visualizar o fluxo do trabalho. De uma forma objetiva, deve ser possível avaliar o estado atual do projeto e entender em qual etapa do fluxo cada parte dele está. Com isso, as pessoas devem poder reconhecer impeditivos e gargalos no sistema de trabalho e atuar para melhorá-lo, de um ponto de vista tático.

Além da visualização do fluxo do trabalho, o Kanban propõe também limitar a quantidade de trabalho em andamento. Essa restrição é chave para consolidar um fluxo de trabalho sustentável. Imagine um cenário em que há três desenvolvedores em uma equipe, e eles fazem um acordo de que a quantidade máxima de tarefas em andamento será quatro. Nesse caso, quando houver pelo menos uma tarefa aguardando revisão e algum desenvolvedor concluir o que estiver fazendo, ele não poderá puxar outro *card* até que revise as alterações que estiverem aguardando revisão. Desse modo, o sistema força que as tarefas sejam movidas em direção à área mais à direita do quadro, impulsionando eficiência de fluxo.

Essa restrição contribui para estabelecermos uma condição de trabalho mais calma porque há menos caos acumulado em forma de trabalho inacabado. Equipes mais maduras inclusive consideram como parte do fluxo do trabalho o monitoramento das tarefas entregues após sua liberação. Elas se sentem responsáveis por coletar e compreender as consequências do trabalho entregue antes de puxar mais tarefas para a etapa de desenvolvimento e, por isso,

asseguram um ritmo mais saudável ao mesmo tempo em que assimilam feedbacks.

Dessa forma, a gestão do fluxo de trabalho é parte crucial para a sustentabilidade do trabalho. O Kanban não serve apenas como uma abordagem que produz maior eficiência de fluxo, mas, também, como uma abordagem mais humana.

Porém, ela não é suficiente. À medida que as aplicações ficam maiores e mais complexas, e as equipes de desenvolvimento crescem, a dor sobre a gestão do conhecimento começa a ficar mais evidente. O software não cabe mais na cabeça dos desenvolvedores — eles não sabem mais onde estão as coisas que precisam e como elas funcionam e, por causa disso, não conseguem evoluir o produto com a mesma velocidade que tinham antes.

4.3 A carga cognitiva derivada da arquitetura do software

Essa é uma história que a maioria das pessoas desenvolvedoras já vivenciaram. Um produto surgiu e a equipe foi capaz de implementá-lo com agilidade. Depois de um tempo, com o produto funcionando para vários usuários, e, portanto, com um legado que precisa ser considerado, a equipe não é mais capaz de manter a mesma velocidade.

As novas melhorias do produto tornam-se mais burocráticas e complexas para serem implementadas, porque elas não podem quebrar funcionalidades antigas. O sistema como um todo agora possui uma quantidade maior de estados possíveis — mais modelos diferentes, mais serviços interdependentes, mais configurações, maior volume de dados etc.

A equipe se sente congelada, com medo de alterar um código, o qual já não entende mais como funciona. As pessoas não conseguem ter coragem de assumir decisões porque não podem prever as consequências das mudanças.

Para endereçar esses problemas, uma série de boas práticas sobre organização de código começou a ser proposta. Essas boas práticas sugerem convenções como: classes com responsabilidade única; baixo acoplamento; métodos pequenos de fácil leitura; e declaração de variáveis com nomes intuitivos.

As pessoas se deram conta de que a maneira como elas escreviam o código determinava quão fácil seria alterá-lo no futuro — e que isso com certeza aconteceria. Nesse cenário, não basta o código funcionar. Ele precisa ser feito com qualidade para que outras pessoas possam entendê-lo, e não apenas máquinas.

A reflexão sobre como o código deve ser escrito para favorecer o entendimento das pessoas se dá em várias camadas, desde a declaração de uma variável em uma função, até a organização das classes, módulos e repositórios. À medida que as áreas de um produto vão ganhando contornos mais claros, as equipes começam a se formar em torno delas, e isso também se reflete no código produzido.

Neste caso, a organização das equipes se modifica de acordo com os pilares de entrega de valor da companhia. Por exemplo, quando há uma aplicação monolítica, equipes buscam reorganizar o código por domínios de negócio, modularizando a aplicação para que esta favoreça seu entendimento e manutenibilidade, melhorando o fluxo das atividades. Outras equipes separam a aplicação em serviços menores e elas adquirem a liberdade (e responsabilidade) de gerenciar de forma independente todo o *pipeline* de implementação e *deploy* daquela parte específica do software.

A necessidade de reorganização do software surge não porque ele não atende aos requisitos de negócio no momento, mas porque ele

não cabe na mente dos desenvolvedores que precisam evoluí-lo.

O processo caótico de implementação de um novo produto nas suas fases iniciais é importante. É bem provável que muito código seja descartado neste processo e que, durante essa trajetória, uma compreensão mais profunda sobre o problema a ser resolvido seja adquirida pela organização. Entretanto, pela sustentabilidade do esquema de trabalho, e pela sanidade das pessoas, reorganizações no software precisam acontecer conforme o produto ganha contornos mais claros.

Quando essas reorganizações não acontecem, a satisfação das pessoas é profundamente afetada. A carga cognitiva para gerenciar uma quantidade insustentável de complexidade em diferentes contextos impede que as pessoas trabalhem com autonomia e responsabilidade, o que reduz o engajamento.

Riscos sobre a implementação de melhorias na arquitetura do software

No momento em que as equipes de desenvolvimento decidem que uma reorganização precisa ser implementada em nível de equipes, processos e arquitetura de software, há duas categorias de riscos que podem prejudicar esta evolução:

1) Quando lideranças não acreditam que essas necessidades são importantes e precisam ser atendidas; 2) Quando as práticas são empurradas goela abaixo dos desenvolvedores.

O primeiro caso se refere a companhias que estão sempre olhando para o curtíssimo prazo. Quando estamos completamente focados nos objetivos da semana seguinte, ou do mês seguinte, é difícil justificar uma melhoria que vai beneficiar a organização no médio ou longo prazo, especialmente quando o *retorno sobre o investimento* é difícil de calcular.

Há motivos válidos para uma empresa se comportar desta maneira — uma empresa pode até mesmo deixar de existir no mês seguinte,

caso alguma entrega não aconteça. Entretanto, é a qualidade do sistema de trabalho que determinará se os grandes objetivos da empresa serão alcançados no médio e longo prazo.

> *"Você não sobe para atingir o nível de seus objetivos. Desce ao nível de seus sistemas."* – James Clear [5]

Essa citação de James Clear possui um contexto individual, relacionado à nutrição de bons hábitos. Entretanto, ela comunica a essência do que precisamos perseguir como equipes que desenvolvem software. Não são os objetivos ousados da companhia que geram bons resultados, mas a qualidade do seu sistema de trabalho.

Por isso, dado que evoluir o sistema de trabalho requer investimento, quando as lideranças da empresa não estão convencidas de que este é um trabalho importante, há um risco para a sustentabilidade do negócio.

Quando você se encontrar nesta situação — precisando convencer seus superiores de que uma refatoração ou melhoria de arquitetura precisa ser implementada para que o sistema de trabalho possibilite mais produtividade —, uma dica é argumentar do ponto de vista de negócio:

- Quanto tempo pode ser reduzido no *ramp up* de novas desenvolvedoras?
- Como o investimento pode afetar o *lead time* médio de tarefas?
- Como poderia afetar a incidência de bugs?
- Como o alto acoplamento existente hoje na aplicação dificultou uma entrega recente? Qual foi prejuízo financeiro disso?

Essas reflexões serão boas, inclusive, para a própria equipe de desenvolvimento avaliar até onde ela deve ir, considerando as condições existentes no negócio — e isso também nos conecta com o segundo risco.

O segundo risco aparece quando o oposto acontece: em vez de nada ser feito, a equipe decide implementar mudanças drásticas na arquitetura imediatamente. Líderes técnicos buscam implementar uma metodologia de desenvolvimento à risca, repensando toda a base de código, porque acreditam que encontraram o jeito certo de desenvolver software. Com isso, pensam que poderão endereçar a todos os problemas que possuem de uma vez. Desta maneira, eles não usam as abordagens como ferramentas, e sim como modelos definitivos.

Quando isso acontece, líderes técnicos não avaliam questões fundamentais da decisão:

- A equipe entende o propósito da mudança?
- A equipe concorda que o problema é relevante?
- A equipe concorda que a nova abordagem vai resolver os problemas?
- A equipe está disposta a adotar a nova arquitetura?
- A equipe terá tempo de aprendê-la?

Além da relação com a equipe, outras premissas mais abrangentes devem ser colocadas à prova:

- Por que acreditamos que a proposta pode reduzir a carga cognitiva e melhorar o fluxo de trabalho?
- Quais são os riscos associados às mudanças? Como elas podem dar errado?
- Existem alternativas para reduzir o escopo da proposta e entregá-la de maneira gradual?

Desafiando as premissas, as equipes poderão tomar decisões melhores e com mais lucidez.

Implementar práticas e metodologias à risca, desconsiderando as especificidades da organização — como as características dos usuários, do produto e das pessoas que o desenvolvem — será sempre um erro. Não há motivos para acreditar que alguém foi capaz de escrever instruções precisas para que sua equipe, neste

exato momento e conjuntura, pudesse segui-las e, assim, ter seus problemas plenamente resolvidos.

> *"O Spotify não implementou o modelo Spotify copiando o Spotify. Por que pessoas em empresas pensam que podem implementar o modelo Spotify copiando o Spotify?"* – Kent Beck [6]

As metodologias devem servir como ferramentas e se adaptarem ao seu contexto, e não o contrário. Especialmente, dado que buscamos novas abordagens e metodologias justamente para estabelecer um fluxo de trabalho humanamente sustentável, não faz sentido implementá-las desconsiderando as necessidades humanas.

Nesta primeira parte do livro, falamos muito sobre a natureza do trabalho do conhecimento e do desenvolvimento de software, e como estes aspectos são primordiais quando almejamos sustentabilidade humana dentro das empresas.

No entanto, esse recorte não é suficiente para pensarmos sobre saúde mental na indústria de software. É possível que uma equipe compreenda todas estas características, aplique uma gestão apropriada e, mesmo assim, pessoas podem estar submetidas à condição de *burnout*.

Para ampliar nossa visão sobre o assunto, na próxima parte olharemos para a cultura das empresas e o espírito do nosso tempo.

Resumo

- Os valores do *Extreme Programming* são centrados nos seres humanos. A metodologia surgiu na busca de reconciliar humanidade e produtividade com valores como comunicação, simplicidade, feedback, coragem e respeito;
- O limite sobre o trabalho em progresso, que é uma das principais práticas do Kanban, favorece a sustentabilidade humana das equipes ao reduzir o estresse gerado pelo acúmulo de trabalho inacabado;
- Quando empresas crescem, elas precisam investir na arquitetura do seu software para manter a eficiência de fluxo das equipes. Caso contrário, a carga cognitiva sobre os desenvolvedores será insustentável.

Referências

[1] BECK, Kent; ANDRES, Cynthia. *Extreme Programming Explained*. 2. ed. Addison-Wesley Professional, 2004. p. 3.

[2] *Ibidem*, p. 21.

[3] *Ibidem*, p. 21.

[4] ZEIGARNIK, Bluma. Das Behalten erledigter und unerledigter Handlungen, *Psychologische Forschungen*, v. 9, p. 1-85. Alemanha, 1927.

[5] CLEAR, James. *Hábitos atômicos*: um método fácil e comprovado de criar bons hábitos e se livrar dos maus. Rio de Janeiro: Alta Books, 2019.

[6] BECK, Kent. 25 mai 2018. (Tweet) Spotify didn’t

implement the Spotify model by copying Spotify. Why do folks at other companies think they can implement the Spotify model by copying Spotify? 25 maio 2018. *Twitter*: @kentbeck. Disponível em: https://twitter.com/kentbeck/status/1000011818197188610. Acesso em 16 set. 2022. Tradução do autor.

PARTE 2 - A EMPRESA E SUA CULTURA

Capítulo 5
Por que o trabalho está uma loucura?

A loucura se tornou o senso comum em várias empresas de tecnologia. Quando eu decidi estudar desenvolvimento de software, lembro que as pessoas comentavam: "é uma profissão promissora, mas eu não aguentaria o estresse". Eu não conseguia entender o motivo, porque parecia muito prazeroso construir produtos e algoritmos, e eu me sentia bem fazendo isso.

Com o tempo, eu entendi do que se tratava. Programar era apenas uma parte do trabalho — lidar com interdependências e expectativas em meio a incertezas tornava o trabalho muito mais complicado. Como vimos na primeira parte deste livro, um descompasso em lidar com essas condições acaba por comprometer a sustentabilidade do trabalho.

Entretanto, as longas jornadas de trabalho com incessantes interrupções não se justificam apenas por essas condições da natureza do desenvolvimento de software. Trata-se de um aspecto da cultura das organizações: a exaustão passou a significar uma medalha de honra. Trabalhar até tarde e estar engajado em várias iniciativas ao mesmo tempo é visto como algo positivo para as empresas, e esse comportamento é recompensado financeiramente. Essas pessoas incorporam o arquétipo do herói e trabalham para salvar projetos a qualquer custo, porque de fato possuem um alto senso de responsabilidade e, com isso, são promovidas. Essa lógica se retroalimenta para o indivíduo — se não estamos exaustos, não estamos fazendo o suficiente.

Esse fenômeno não é algo particular das empresas ou equipes que desenvolvem software. É algo mais abrangente, que está intrínseco na maneira de pensar dos nossos tempos.

"Ralar enquanto os outros dormem. Isso é empreender."

Essa citação é de uma página para empreendedores. Ao que parece, para obter sucesso você deve se privar de sono para trabalhar, caso contrário, você não está empreendendo. Claramente, isso não é sustentável. A privação do sono, além da fadiga e irritabilidade no curto prazo, pode levar a problemas mais sérios à saúde no longo prazo, como obesidade, doenças cardíacas e diabetes [1].

Mas, como chegamos até aqui? Como podemos pensar que dormir menos para trabalhar seria uma boa ideia?

5.1 Como mudamos nossa relação com o trabalho ao longo dos tempos?

Como Rahaf Harfoush descreve em seu livro *Hustle & Float: Reclaim Your Creativity and Thrive in a World Obsessed with Work*, nossa relação com o trabalho se transformou muito ao longo dos tempos.

Na Grécia Antiga, a ausência de necessidade de trabalhar era uma das coisas que distinguia as classes sociais. A elite teria o privilégio de ter tempo para lazer, arte e filosofia. Portanto, buscava-se a ausência de trabalho — não existia a economia do conhecimento. Na Idade Média, com a ascensão do Cristianismo, o significado do trabalho passou de uma punição divina para, depois da Reforma Protestante, uma dádiva — uma maneira de servir a Deus.

Apenas há pouco tempo em nossa história, no início do século XX, surge o conceito do *Sonho Americano*. O termo foi cunhado pelo historiador James Truslow Adams em 1931 e traz a ideia de que, independente de sua classe social de nascimento, você seria capaz

de alcançar qualquer estatura social, desde que trabalhasse duro para isso. Apesar de uma visão excessivamente romantizada quando comparada com a realidade, as estruturas sociais de fato se organizaram de uma maneira sem precedentes em termos de mobilidade social.

No final do século XX, o espírito de nosso tempo se transformava novamente, assim como nossa relação com o trabalho. O sociólogo e filósofo polonês Zygmunt Bauman foi uma das pessoas que mais bem conseguiu articular os sinais das mudanças que aconteciam no final dos anos 90. Ele descreve nosso tempo como *modernidade líquida*, devido ao derretimento dos sólidos que eram suspeitos de, certo ou errado, limitarem nossa liberdade individual [2]. O derretimento representa o enfraquecimento das instituições, como a influência de igrejas, governos e mídias centralizadas.

Com isso, há uma ênfase na individualização. Hoje, os indivíduos são responsáveis por definirem seus próprios objetivos de vida e usarem seus próprios recursos para alcançá-los. Assim, houve uma ressignificação das profissões, e a busca individual por identidade passou a se dar principalmente pelo trabalho.

Somos o que fazemos

Quando nos apresentamos para pessoas estranhas, é muito comum primeiramente falarmos sobre nossas profissões e onde trabalhamos. No âmbito do desenvolvimento de software, falamos também sobre as tecnologias e *frameworks* de que gostamos mais, especialmente no início de nossas carreiras. Esses atributos, de alguma forma, definem quem somos.

Portanto, não é mais suficiente que nosso emprego cubra nossas necessidades básicas e de bem-estar; ele também deve satisfazer o propósito que atribuímos a nós mesmos. Desta maneira, estamos intimamente conectados com nossa profissão.

O pensamento do *Sonho Americano* ainda se preserva nos nossos tempos, mas de maneira aprimorada. Agora, não basta que o sucesso seja alcançado — ele precisa chegar rápido. As pessoas buscam produzir conteúdos na internet para viralizar, e as empresas buscam desenvolver softwares para crescer exponencialmente e serem vendidas em seguida no menor período de tempo possível.

Entretanto, quando temos nossa identidade tão conectada com o que fazemos como profissão, o que acontece quando isso não acontece? Em especial, o que acontece quando fracassamos e o espírito de nosso tempo diz que somos inteiramente responsáveis por isso — mesmo quando estamos inseridos no mundo de incertezas da economia do conhecimento? Emoções fortes aparecerão, e elas influenciarão a maneira como agimos.

5.2 Longas jornadas

Diferente do que acontecia na Grécia Antiga, quando o lazer e ócio eram objetivos das classes mais privilegiadas, hoje, estar desocupado significa irrelevância. Muitos de nós nos sentimos culpados quando não estamos no nosso limite e buscamos *hacks* de produtividade para podermos cumprir com todos os compromissos.

Procurar melhorar a própria maneira de trabalhar para ser mais produtivo é ótimo. Mas, otimizar cada minuto do dia como se fôssemos máquinas não é a solução. Um computador, mesmo com apenas uma CPU, pode lidar bem com todas as suas tarefas e trocas de contexto. Ele será utilizado ao máximo enquanto otimiza o tempo de *handoff* dos processos — quando uma tarefa estiver aguardando por *IO*, o processador poderá dedicar mais tempo para outras tarefas. Porém, o ser humano não funciona como um computador.

Nós temos um limite de energia que pode ser consumido ao longo do dia em forma de atenção e autocontrole — aspectos necessários

para entregar um bom trabalho criativo. Como o psicólogo e pesquisador Roy Baumeister demonstrou em uma série de experimentos, tarefas que demandam esforço cognitivo, emocional e físico (ou seja, atividades que exigem autocontrole) compartilham, pelo menos em parte, de uma mesma "fonte de energia" [3].

Quando organizamos nossa semana pulando de um compromisso para outro, além de desconsiderar a fonte limitada de energia que possuímos para um bom dia de trabalho, há um resíduo na forma de estresse que compromete a qualidade das nossas escolhas ao longo do tempo.

Trabalhar demais é ruim, mas por que não conseguimos parar?

Muitas empresas já entendem a importância do descanso para a produtividade e crescimento sustentável da companhia, mas por que não conseguimos parar?

Mesmo as empresas oferecendo benefícios que possibilitem aos colaboradores estar mais tempo fora do trabalho, as pessoas ficam receosas em utilizá-los. Há algo no ar que gera uma expectativa de que devemos chegar cedo e sair tarde, e que precisamos estar disponíveis a qualquer momento.

Em 2020, especialmente, a pandemia de COVID-19 forçou várias empresas a trabalharem de maneira remota de um dia para o outro, e as pessoas ficaram confusas com o que esperavam delas. Dado que eu estou trabalhando da minha casa, devo responder mensagens o mais rápido possível e estar presente em todas as reuniões para provar que estou trabalhando?

Nesse momento, há um conflito entre as políticas da empresa que incentivam que as pessoas tenham mais tempo de qualidade fora da companhia, com a cultura de devoção ao trabalho e estas expectativas que não estão escritas. Não adianta existirem benefícios como horário flexível, licença maternidade e paternidade

estendidas, férias remuneradas, se as pessoas não sentem que podem usufruí-los sem culpa.

Como Jason Hall escreve em seu artigo intitulado *Why Men Don't Take Paternity Leave* (Por que homens não tiram licença paternidade), publicado na Forbes em 2013, muitos pais sentem que há um estigma em ser a pessoa que aproveita o período máximo de licença paternidade — não apenas por medo de perderem seus empregos, mas também pela desaprovação de seus colegas e questionamentos sobre seu compromisso e dedicação [4].

No Japão, um dos países mais generosos em termos de licença maternidade e paternidade, o período de afastamento pode chegar a um ano para mães e pais. Entretanto, em 2014, apenas 2.3% dos pais elegíveis utilizaram o benefício, em grande medida por causa do conflito em agir contra as normas sociais de masculinidade [5].

Já para mulheres, há a percepção de que é muito difícil a conciliação entre uma carreira profissional e a construção de uma família, e elas são pressionadas a escolher entre uma ou outra. Uma pesquisa publicada pela Fundação Getúlio Vargas em 2016 demonstrou que a probabilidade de emprego de mães no mercado de trabalho cai substancialmente após a licença maternidade. Os efeitos são heterogêneos com relação ao nível educacional da mãe: em 12 meses, trabalhadoras com maior escolaridade apresentam queda de emprego de 35%, enquanto mulheres com nível educacional mais baixo apresentam 51% de queda [6].

Portanto, mesmo com o motivo de querer construir uma família e poder estar presente no primeiro ano de vida de um filho ou filha, a cultura de devoção ao trabalho impede que as pessoas se afastem de suas responsabilidades profissionais. A existência dos benefícios é fundamental, mas ela não é suficiente. Precisamos desconstruir as expectativas desumanas de disponibilidade no trabalho para estabelecermos um ritmo mais sustentável, que nos permita ter energia para fazer o que realmente importa, dentro e fora do trabalho.

Como a cultura workaholic se reforça nas empresas

Há muitos motivos que justificam jornadas prolongadas de trabalho. Quando projetos começam a atrasar e expectativas não são atendidas, por exemplo, pessoas se movem para fazer o que for necessário para atender aos objetivos colocados, mesmo que isso signifique fazer horas extras.

Essas ações nem sempre são motivadas por compensações financeiras. As pessoas assumem mais responsabilidades para mitigar um risco do grupo porque desejam colaborar — como evitar a perda de um cliente, perseguir resultados da companhia ou apagar incêndios operacionais. Entretanto, esses profissionais normalmente recebem um merecido reconhecimento financeiro por estas atitudes. Eles ampliaram suas capacidades para colaborar com os objetivos da empresa e, por causa disso, recebem compensações financeiras pelo desenvolvimento.

Porém, há duas consequências que podem surgir como resultado desta dinâmica:

- Problemas de gestão são encobertos;
- A mentalidade *workaholic* é disseminada.

No primeiro caso, como o projeto foi entregue com sucesso e atendeu de maneira satisfatória os anseios de todas as partes, não há uma reflexão sobre as falhas de gestão que levaram as pessoas a fazerem horas extras. Na verdade, já que o projeto foi implementado em uma velocidade excepcional, este pode ser considerado um exemplo de projeto bem conduzido — os gerentes foram capazes de levar a equipe ao limite para atender o prazo estabelecido.

No livro *Peopleware: Productive Projects and Teams*, Tom DeMarco descreve uma conversa que teve com um gerente de projetos do Sul da Califórnia. O gerente relatava as consequências da quantidade insana de horas que sua equipe empregava — como pelo menos dois divórcios, que aconteceram durante o andamento de um

projeto, estariam diretamente relacionados às horas extras impostas. À medida que a conversa continuava, Tom percebeu que esse gerente não estava se queixando das situações, ele estava se gabando pelos seus métodos [7].

Esse livro foi escrito em 1987, mas ainda podemos observar este tipo de visão em gerentes de projetos de software.

> *"Apesar de toda a conversa sobre 'trabalhar de maneira mais inteligente', há uma sensação generalizada de que gestão no mundo real é sobre fazer pessoas trabalharem mais duro e por mais tempo, em grande parte às custas de suas vidas pessoais."*
> – Timothy Lister e Tom DeMarco [8]

Além de os problemas de gestão serem encobertos quando as pessoas salvam os projetos, desta maneira, a mentalidade *workaholic* é reforçada. As pessoas que abrem mão do final de semana e trabalham até tarde são as que são promovidas e recompensadas. A mensagem que fica para os demais é de que este é o caminho para crescer dentro da empresa.

Mas essa não é uma cultura sustentável. Eventualmente as pessoas perceberão às custas de que o crescimento profissional está acontecendo e avaliarão o que realmente importa. Assim que as pessoas perceberem que sacrificaram um valor mais importante, como a família, por um menos importante, como o trabalho, será devastador para elas. Esse profissional buscará alternativas em outras companhias.

Portanto, as empresas devem valorizar o senso de responsabilidade dos seus colaboradores de maneira que isso não se coloque em detrimento dos seus valores pessoais mais importantes. Isso pode ser feito ao reconhecer o crescimento deles como profissionais (novas habilidades adquiridas, maior autonomia e boa articulação entre equipes) e as contribuições da equipe de que faz parte (como

resultados conquistados e aprendizados obtidos). Tudo isso com o exemplo das lideranças de que os objetivos não devem ser perseguidos a qualquer custo — eles não devem se sobrepor a valores inegociáveis que estão fora do trabalho.

Isso pode implicar em resultados piores em curto prazo, mas é fundamental quando buscamos construir um negócio que perdure. Nesse caso, as empresas devem se responsabilizar por criar um ambiente em que seus colaboradores possam trabalhar por anos e crescer na carreira, sem consequências negativas para sua saúde e relacionamentos pessoais.

O que significa dar o nosso melhor

Quando nos enxergamos como máquinas, dar o nosso melhor significa utilizar o máximo de nossos recursos. Mas, de maneira analítica, podemos otimizar apenas o que medimos. Nesse caso, no trabalho do conhecimento, nosso recurso mais importante que somos capazes de medir é o tempo.

Como não somos capazes de medir a nossa "fonte de energia para autocontrole e concentração", nós a ignoramos. Buscamos otimizar o nosso tempo ao máximo: cada segundo do dia é preenchido visando a produtividade. Mesmo estando exaustos no fim do dia, o tempo é nosso tirano — se ainda não é hora de parar, devemos seguir em frente, não importa o que nosso corpo esteja nos dizendo.

Na terceira parte deste livro aprofundaremos sobre a gestão de tempo e energia. Por enquanto, precisamos saber que esta maneira de medir a otimização de nossos recursos, preenchendo cada segundo do dia com trabalho produtivo, não é a maneira mais apropriada. A verdade é que dificilmente podemos dar o nosso melhor nessas situações.

Precisamos estar em condições de tomar as melhores decisões, mesmo que elas sejam difíceis. Para isso, descansar é fundamental.

Quando estamos cansados, nossa capacidade de avaliação é duramente comprometida.

Como Daniel Kahneman descreve em seu livro *Thinking Fast and Slow*, um estudo feito com juízes que decidiam sobre liberdade condicional em Israel, publicado na *Proceedings of the National Academy of Sciences*, encontrou uma forte correlação entre o resultado de reprovação do pedido e o tempo desde a última refeição do juiz.

> *"Os autores do estudo plotaram a proporção de solicitações aprovadas em relação ao tempo desde a última pausa para refeição. A proporção aumenta após cada refeição, quando cerca de 65% dos pedidos são atendidos. Durante mais ou menos duas horas até a próxima refeição do juiz, a taxa de aprovação cai continuamente para cerca de zero pouco antes da refeição."*
> – Daniel Kahneman [9]

Com os resultados alarmantes, os autores da pesquisa revisaram cuidadosamente possíveis explicações. Entretanto, a explicação mais plausível encontrada foi a de que juízes cansados e com fome tendem a recair na opção mais fácil, que é a da reprovação do pedido.

Portanto, este fenômeno, que foi chamado de *esgotamento do ego*, influencia fortemente a dinâmica de nosso trabalho, na medida em que somos colocados em situações nas quais precisamos fazer avaliações de maneira subjetiva e com informações incompletas. Por exemplo, qual é a estrutura de equipes mais apropriada para fazer a gestão do produto, dados os recursos e desafios que temos hoje? Qual é a arquitetura de solução que poderia se encaixar de maneira satisfatória para atender os requisitos de negócio? Como devo avaliar as tarefas do quadro Kanban na relação urgência e importância delas?

Nesses casos, não há exatamente uma resposta certa e, para tomar a melhor decisão que podemos, precisamos estar nas condições apropriadas — com energia para fazer as avaliações que precisamos, usando o tempo que for necessário.

Criatividade não pode ser produzida com força bruta

Como já falamos em seções anteriores deste livro, é a qualidade de nossas escolhas que determina nosso progresso, e não a quantidade delas. Entretanto, quando estamos em busca de uma solução, a velocidade de nossas tentativas contribui para nos aproximarmos dela — aprendemos com os feedbacks e a qualidade de nossas decisões melhora com as tentativas. Nesse sentido, um sistema de trabalho que permite um bom fluxo de atividades é crucial para a companhia progredir.

Estabelecer um bom fluxo de trabalho não está relacionado com investir mais horas para fazer mais coisas. O fluxo está relacionado com garantir que as atividades que são iniciadas — que as equipes consideraram as mais importantes e se comprometeram em entregar — sejam finalizadas e um feedback seja obtido.

> *"Hacks para gerenciamento do tempo, hacks para o sono, hacks para o trabalho. Tudo isso reflete nossa obsessão por tentar espremer mais tempo do dia, mas rearranjar os padrões diários para encontrar mais tempo não é o problema. Muita coisa para fazer é o problema."* – David Heinemeier Hansson e Jason Fried [10]

Criatividade está em conseguir fazer menos — encontrar soluções simples e ser capaz de se comprometer com elas, o que significa falar não a outras oportunidades. Técnicas para produtividade são importantes, mas, antes disso, precisamos considerar a finitude da energia do nosso dia para atividades que demandam esforço cognitivo e filtrar o que é realmente essencial. Dessa maneira, as

pessoas estarão em uma posição mais favorável para produzir um trabalho de qualidade, porque não podemos espremer o tempo para extrair boas decisões.

5.3 Protegendo a atenção

Na seção anterior, falamos sobre como as longas jornadas de trabalho podem estar ligadas a crenças mais abrangentes da nossa cultura, e como a falta de um ritmo sustentável pode impactar a nossa eficácia como profissionais e empresas, na medida em que a qualidade de nossas decisões são afetadas.

Porém, é possível que as pessoas estejam chegando cedo e saindo tarde por outro motivo: elas simplesmente não conseguem realizar o trabalho no horário normal. Devido a incessantes interrupções, as pessoas não conseguem manter o período prolongado de atenção necessária que um bom trabalho criativo demanda. Por causa disso, buscam começar quando ainda não há ninguém trabalhando e ficar até mais tarde, quando todos já foram embora.

Entretanto, como já falamos em capítulos anteriores, sempre haverá interdependência no trabalho de desenvolvimento de software. Projetos de software, quando crescem, tendem a se tornar excessivamente complexos para que apenas uma pessoa seja capaz de evoluí-lo de ponta a ponta — desde a compreensão sobre o que precisa ser feito, até o desenvolvimento de todas as suas camadas e liberação do produto.

Então, como lidamos com esta contradição? Precisamos proteger a atenção das pessoas, mas, ao mesmo tempo, precisamos resolver as interdependências e *handoffs* para que o trabalho possa fluir. Para lidar com esse dilema, precisamos refletir sobre dois aspectos:

- Como as equipes estão organizadas;
- Como as pessoas se comunicam.

Como a divisão de equipes promove ou impede o fluxo do trabalho

Há inúmeras maneiras de organizarmos equipes que desenvolvem software. Pense, por exemplo, na quantidade de funcionalidades que uma rede social possui: as pessoas podem fazer publicações para interagir com suas conexões; os usuários devem receber notificações para manter o engajamento; e um algoritmo deve determinar qual o conteúdo mais interessante para ser apresentado em seguida na *timeline* dos usuários.

Além disso, há vários *stakeholders* em contato com o produto: o usuário que produz e consome conteúdos do seu interesse; a empresa que deseja fazer anúncios e obter retornos financeiros; usuários da própria empresa que desenvolve a rede social e que realizam tarefas de *back office* para que a operação funcione. Neste contexto, existem milhares de casos de uso, e diferentes usuários, com objetivos distintos, interagindo com o software.

Ao mesmo tempo, há as especialidades inerentes do desenvolvimento do software. Para o produto ser entregue e continuar funcionando, pessoas precisam: determinar fluxos de interação; desenhar interfaces; implementar as várias camadas do software (front-end, back-end); desenvolver algoritmos que sugerem conteúdos baseado em dados coletados previamente; gerenciar *pipelines* de *deploy*; cuidar da infraestrutura; implementar mecanismos de segurança para proteção dos dados, entre várias outras atividades específicas.

Portanto, como devemos organizar as equipes de maneira que favoreça a fluidez do trabalho que precisa ser entregue e proteja a atenção das pessoas? Essa reflexão é crucial, pois a maneira como as equipes são organizadas determina muito sobre o resultado final — as interações fortalecidas e enfraquecidas podem permitir ou impedir o emergir de soluções.

No exemplo anterior, podemos pensar a organização no clássico modelo industrial do trabalho. Nesse caso, a divisão das equipes se dá pelas especialidades de técnicas: front-end, back-end, ciência de dados, infraestrutura, banco de dados, segurança etc. Por consequência, haverá uma enorme interdependência no processo, porque várias equipes precisarão se mover de maneira coordenada para entregar uma funcionalidade, repassando trabalho de uma para outra.

Uma abordagem mais apropriada seria separar as equipes nos pilares de entrega de valor do produto. Imagine uma empresa que oferece um serviço de streaming de conteúdo. Esse software provavelmente precisará de funcionalidades como: um fluxo de assinatura e cobrança; um algoritmo de recomendação de conteúdos; um controle de catálogo. Todas essas partes demandam um ferramental distinto para atender a necessidades de negócio que são diferentes entre em si, mas que são fundamentais para o negócio como um todo — a assinatura deve ser um processo seguro e fácil; as recomendações devem ser alinhadas aos interesses do usuário; e o controle de catálogo deve permitir que a operação da companhia funcione de maneira ágil e segura. Faz sentido que as pessoas sejam organizadas em grupos com o objetivo de cuidar dessas partes de maneira independente.

É muito provável que a equipe de recomendação de conteúdos demande pessoas com conhecimento em estatística, por exemplo, mas isso não deve ser verdade para a equipe que cuidará do catálogo de conteúdos. Por outro lado, ambas as equipes devem demandar engenheiros de software com experiência em back-end e banco de dados.

Dessa maneira, as equipes podem progredir da maneira mais autônoma — ao entregar melhorias com o mínimo de *hand offs* com outras equipes, já que elas são autossuficientes em termos de conhecimento. A interdependência dentro da equipe será fortalecida, porque continuará havendo especialistas para lidar com os desafios técnicos de cada camada do software, mas uma equipe não

dependerá de outras equipes (ou dependerá pouco) para entregar o software para os usuários.

Podemos traçar um paralelo desta ideia com os princípios de alta coesão e baixo acoplamento de Robert C. Martin. O autor do *Clean Code* propõe que, para mantermos um código simples e fácil de se fazer manutenção, as classes devem ter responsabilidades únicas (não devem assumir responsabilidades que não são suas) e depender o mínimo possível de outras classes (manter baixo acoplamento). Da mesma maneira, as equipes devem ter alta coesão nos limites das suas responsabilidades e baixo acoplamento com outras equipes para mover-se com agilidade e autonomia.

Precisamos reconhecer que há um limite na carga cognitiva que uma pessoa ou equipe suporta, e, portanto, o escopo de suas responsabilidades deve ser ajustado para caber nesta capacidade. Assim, gerenciamos a carga cognitiva e reduzimos a troca de contextos.

> *"Fluxo rápido requer restringir a comunicação entre times. Colaboração de equipes é importante para áreas cinzentas do desenvolvimento, onde descoberta e especialidade são necessárias para fazer progresso. Mas, em áreas onde a execução prevalece — não descoberta — comunicação se torna uma sobrecarga desnecessária."* – Manuel Pais e Matthew Skelton [11]

Como Manuel Pais e Matthew Skelton argumentam no livro *Team Topologies: Organizing Business and Technology Teams for Fast Flow*, uma comunicação eficaz não é uma em que todos sabem o que se passa em todas as equipes. Precisamos promover as interações necessárias e evitar as desnecessárias — garantir que a companhia se mova para um objetivo em comum, mas gerenciando a sobrecarga cognitiva de alinhamento. Quando a companhia tem uma expectativa de que todos devem ler as mensagens no chat, ou

de que todos precisam estar presentes em reuniões para aprovar decisões, isso indica um problema de design organizacional.

Boas práticas para interrupções internas

Apesar de nos organizarmos como equipes de modo a evitar a interação indesejada, interrupções invariavelmente vão acontecer. Por isso, precisamos pensar em boas práticas para essas situações. Uma boa premissa para guiar nossas decisões sobre como agir nesses casos é: considere que a outra pessoa está fazendo um trabalho tão importante quanto o seu.

> *"Quando alguém pega seu tempo, isso não custa nada a elas, mas custa tudo a você."* – David Heinemeier Hansson e Jason Fried [12]

Como David Heinemeier Hansson e Jason Fried escrevem em seu livro *It Doesn't Have to Be Crazy at Work*, e como já falamos em seções anteriores, há um custo nas interrupções. O estado de fluxo necessário para o trabalho criativo acontecer é interrompido e precisamos colocar mais energia para reestabelecer a concentração e produtividade. Quando interrompemos alguém para obter alguma informação de que precisamos, este custo fica completamente com a pessoa, enquanto eu obtenho o que eu desejo.

Escritórios abertos, pensados para facilitar a comunicação das pessoas dentro da empresa, são bons para promover interação. Porém, ao mesmo tempo, eles promovem distração e reduzem a privacidade das pessoas. Nesse contexto, faz sentido que os membros de uma equipe estejam fisicamente próximos, para facilitar os rituais e viabilizar programação em pares, por exemplo. Entretanto, no caso de uma equipe de vendas ao lado de uma equipe de desenvolvimento, a troca entre distração e interações valiosas pode não valer a pena.

A ideia central ao pensarmos em boas práticas de comunicação, que buscam evitar interrupções, é mover a maior quantidade possível de interações para ferramentas assíncronas, o que permite que a outra pessoa gerencie o momento mais apropriado para reagir a elas. Antes de impor um custo de interrupção para um colega de trabalho, faça a seguinte reflexão:

- Esta ligação poderia ser uma mensagem de texto?
- Esta mensagem de texto poderia ser um e-mail?
- Este e-mail precisa ser enviado para todas estas pessoas?
- Este e-mail precisa ser enviado?

Evidentemente, urgências que demandam uma ação rápida vão aparecer. Mas a verdade é que pouquíssimas situações realmente exigem uma interrupção imediata — que não podem esperar por algumas horas ou dias. Sabendo disso, podemos estabelecer um ambiente mais calmo e satisfatório, onde as pessoas podem entregar o melhor que podem no fim do dia.

Algumas pessoas, seja por seu estilo pessoal ou um momento específico de um projeto, podem preferir realizar comunicações síncronas com mais frequência. Quando ideias precisam ser discutidas e elas ainda não estão tão bem estabelecidas, é mais difícil articulá-las por texto em grandes *threads* de e-mail ou mensagens. Nesses casos, faz sentido promover mais reuniões para que o feedback trocado desbloqueie ou oriente os próximos passos das pessoas de maneira mais assertiva.

Quando os rituais de alinhamento já existentes não são suficientes para atender a essas necessidades, como as reuniões diárias ou reuniões de *sprint* semanais, as pessoas podem fazer acordos para que estejam disponíveis quando necessário por um período específico do dia. Por exemplo, os membros de uma equipe podem acordar de trabalharem de maneira focada no período da manhã, com o mínimo de interrupções, mas de maneira mais aberta a interrupções à tarde.

O trabalho está uma loucura nas empresas, não apenas pelos aspectos inerentes à atividade de desenvolvimento, como complexidade e incertezas; há também crenças enraizadas na maneira de pensar do tempo em que vivemos que nos incentiva a ser a pessoa que está sempre disponível, porque nossa identidade está estritamente relacionada com nossa profissão. Nós nos engajamos em todas as iniciativas que podemos, dentro e fora das empresas, e nos sentimos frustrados por não conseguir dar conta, ao mesmo tempo em que percebemos que abrimos mão de valores pessoais mais importantes.

Acreditamos que este é o único caminho para prosperar, mas não precisa ser assim. Podemos desconstruir as crenças por trás das longas jornadas para sermos capazes de entregar um trabalho com qualidade, e podemos fazer isso ao proteger e respeitar o tempo das pessoas — reorganizando equipes, considerando a carga cognitiva necessária para lidar com o software e indicando boas práticas para interrupções.

Apesar disso, quando temos nossa identidade bastante relacionada com nosso trabalho, emoções fortes sempre vão aparecer. Precisaremos ser capazes de dar e receber opiniões sinceras, que nunca serão boas de falar e ouvir. Teremos que ter conversas difíceis, e este será o tema do próximo capítulo.

RESUMO

- O trabalho está uma loucura, não apenas por características inerentes da natureza do trabalho de software. Há aspectos culturais, do espírito do nosso tempo, que influenciam a maneira como agimos. Hoje, há um forte senso de individualidade e identificação pessoal por meio da profissão;
- Longas jornadas de trabalho podem acontecer porque há uma cultura na organização que favorece este comportamento. Problemas de gestão são encobertos por pessoas que trabalham muito mais do que esperado, e salvam projetos, e são recompensadas com promoções;
- Devemos proteger o tempo e atenção das pessoas com a premissa de que elas estão fazendo um trabalho tão importante quanto o nosso. Podemos evitar interações desnecessárias e fortalecer as necessárias ao reorganizar equipes, assim como estabelecer boas práticas de comunicação que reduzem as interrupções.

Referências

[1] The importance of sleep. (https://www.nhsemployers.org/articles/importance-sleep)

[2] BAUMAN, Zygmunt. *Modernidade Líquida.* Rio de Janeiro: Zahar, 2001.

[3] BAUMEISTER, Roy F.; BRATSLAVSKY, Ellen; MURAVEN, Mark; TICE, Dianne M. Ego Depletion: Is the Active Self a Limited Resource? *Journal of Personality and Social Psychology*, v. 74, n. 5, p. 1252-1265. American Psychological Association, 1998. Disponível em: https://faculty.washington.edu/jdb/345/345%20Articles/Baumeister%20et%20al.%20(1998).pdf. Acesso em 16 set. 2022.

[4] HALL, Jason. *Why Men Don't Take Paternity Leave.* Forbes (Website). 2013. Disponível em: https://www.forbes.com/sites/learnvest/2013/06/14/why-men-dont-take-paternity-leave. Acesso em 16 set. 2022.

[5] HARFOUSH, Rahaf. *Hustle & Float*: Reclaim Your Creativity and Thrive in a World Obsessed with Work. Nova Iorque: Diversion Books, 2019.

[6] FUNDAÇÃO GETÚLIO VARGAS (Brasil). *Mulheres perdem trabalho após terem filhos*. (Website). Disponível em: https://portal.fgv.br/think-

tank/mulheres-perdem-trabalho-apos-terem-filhos. Acesso em 16 set. 2022.

[7] DEMARCO, Tom; LISTER, Timothy. *Peopleware*: Productive Projects and Teams. 3. ed. Boston: Addison-Wesley Professional, 2013. p. 13.

[8] *Ibidem*.

[9] KAHNEMAN, Daniel. *Thinking, Fast and Slow*. Nova Iorque: Farrar, Straus and Giroux, 2013. Tradução livre do autor.

[10] HEINEMEIER, David; FRIED, Jason. *It Doesn't Have to Be Crazy at Work*. Nova Iorque: Harper Business, 2018. Edição Kindle, p. 522. Tradução livre do autor.

[11] PAIS, Manuel; SKELTON, Matthew. *Team Topologies: Organizing Business and Technology Teams for Fast Flow.* Portland: IT Revolution, 2019. Tradução livre do autor.

[12] *Ibidem* n. 10, p. 1339. Tradução livre do autor.

Capítulo 6
Conversas difíceis

Como falamos em capítulos anteriores, há duas condições em que estamos inseridos como pessoas desenvolvedoras e que, inevitavelmente, levarão a conversas difíceis no trabalho. Primeiramente, há uma forte interdependência no processo de desenvolvimento de software — estamos a todo momento fazendo acordos e estabelecendo compromissos para que possamos entregar o produto que precisamos. Em segundo lugar, temos nossa identidade como pessoas bastante conectada com nossa profissão. Nossa autopercepção como profissionais está intimamente ligada à nossa autoestima — pensamos que somos o que fazemos. Essa maneira de pensar faz parte do espírito do nosso tempo.

Nessas situações, conversas difíceis precisam acontecer. Na condição de interdependência, precisamos conseguir criticar o trabalho de outras pessoas, e também ouvir críticas, para entregarmos um produto com qualidade. Isso não é uma tarefa simples quando temos nossa identidade bastante conectada com a forma como percebemos nosso valor como profissionais.

Quando há um conflito de opiniões, as discussões podem ficar desconfortáveis porque podemos sentir que, de alguma maneira, estamos ofendendo ou decepcionando as outras pessoas por pensar diferente.

Isso pode acontecer mesmo quando a conversa é sobre um assunto essencialmente técnico — por exemplo, a maneira como um projeto deve ser conduzido ou uma decisão de arquitetura de uma solução. Mesmo nessas situações, na maioria das vezes não haverá uma resposta certa, e sim opiniões pessoais, que precisam ser expostas para que o grupo encontre a maneira de agir que possua maior probabilidade de sucesso para o que se busca resolver.

Em momentos em que um feedback negativo precisa dado a um colega de trabalho, percebemos que somos despreparados para esta tarefa. Por causa disso, é comum evitarmos as conversas difíceis. Preferimos fazer vista grossa para o elefante na sala e não falar sobre algo que nos incomoda para evitar o desconforto.

Mas esse comportamento tem um custo. Quando não falamos aquilo de que precisamos com frequência, um desgaste emocional poderá se acumular ao longo do tempo. Por exemplo, se a maneira como alguém se comunica nos ofende ou acreditamos que esteja ofendendo outras pessoas, e evitamos lidar com essa situação, isso vai se tornar um empecilho para a colaboração, porque, de maneira consciente ou inconsciente, evitaremos a interação com esta pessoa para nos preservar deste desgaste emocional.

Como já vimos, a colaboração é crítica para que projetos de software sejam entregues, e essa maneira de agir, evitando as conversas difíceis, trará consequências negativas para a organização.

Ao mesmo tempo em que conversas difíceis são necessárias, abordá-las da maneira errada pode ser ainda pior. Se temos uma situação incômoda e o resíduo da tentativa de solucionar o problema foi negativo e resultou em sentimentos de raiva e medo, o desgaste emocional cresce mais ainda. Alguém pode ter "vencido" a discussão, mas todos perderam em termos de melhoria na convivência e, por consequência, nas possibilidades de melhorias futuras no sistema de trabalho.

6.1 Por que são difíceis?

Conversas podem ser difíceis por vários motivos. Algumas vezes, podemos pensar que a dificuldade está em lidar com o outro — como quando pensamos "com aquela pessoa, é difícil de lidar". Quando precisamos falar algo para alguém que tem uma

"personalidade forte", ou com quem não nos damos muito bem, a tarefa se torna difícil, por mais simples que seja a mensagem.

Entretanto, não podemos nos eximir desta responsabilidade. Na verdade, na maioria das vezes o impedimento não está no outro, mas em nossa dificuldade de nos expressarmos. Nunca fomos ensinados a comunicar nossos sentimentos e necessidades de maneira eficaz. Ao invés disso, somos treinados desde criança a segurar nossos sentimentos quando emoções fortes surgem. Somos treinados a não chorar quando estamos tristes e a não gritar quando estamos com raiva.

De fato, expressar essas emoções dessas maneiras na maioria das vezes não vai ajudar. Entretanto, deveríamos ser treinados para expressar esses sentimentos e as necessidades não atendidas por trás deles com plenitude, ao invés de escondê-los.

Quando não queremos decepcionar alguém, ou quando temos algum medo da reação, temos ainda mais dificuldade de articular uma mensagem que realmente comunica o que queremos dizer. Nessas situações, é frequente preferirmos não falar nada, ou falar outra coisa, em vez do que precisa ser dito. Agimos para preservar a relação que temos hoje, e não tentamos mudar algo que nos incomoda.

Evidentemente, essa não é uma maneira saudável de nutrir nossas relações. Precisamos ser capazes de manter relações que evoluem ao longo do tempo, tomando cuidado para não acumular questões não resolvidas que levem ao ressentimento.

6.2 A abordagem da Comunicação Não Violenta

Para aprendermos a ter conversas difíceis, traremos neste capítulo a abordagem de Marshall Rosenberg denominada **Comunicação Não Violenta** (CNV) [1]. A CNV busca estabelecer relações de confiança por meio da empatia e se baseia em quatro componentes fundamentais para alcançar uma comunicação eficaz: observação, sentimento, necessidade e pedido. Esses componentes poderiam se encaixar em uma frase da seguinte maneira:

> Quando acontece (observação), sinto-me (sentimento), porque preciso de (necessidade). Portanto, eu gostaria de (pedido).

Aprofundaremos cada um dos quatro componentes nas próximas seções. Por enquanto, considere o seguinte exemplo. Digamos que um líder técnico faz o seguinte feedback para uma desenvolvedora: "Você deve entregar código com mais qualidade". Esse é um feedback ruim porque muitas questões fundamentais não estão explícitas na mensagem. A desenvolvedora deverá se perguntar:

- O que é qualidade para ele?
- Por que ele está falando isso — estou escrevendo código ruim?
- Como eu posso escrever código com mais qualidade, dadas as restrições de tempo do projeto em que estou inserida?
- Por que eu devo entregar mais qualidade — isso é o que falta para eu ser promovida?

Uma maneira mais apropriada de comunicar o feedback, seguindo a estrutura proposta pela CNV, seria a seguinte:

> Quando você entrega funcionalidades sem cobertura de testes, eu me sinto inseguro, porque preciso confiar na suíte de testes do projeto para termos agilidade nas entregas sem perder qualidade. Você poderia garantir uma cobertura de testes que contemple os casos de uso principais das próximas funcionalidades que você desenvolver?

Dessa maneira, a comunicação é muito mais clara. O líder está pontuando especificamente a falta de cobertura de testes do código que a desenvolvedora entregou. Ele também assumiu responsabilidade sobre seu sentimento de insegurança e reforçou as necessidades por trás dele — caso o projeto não possua uma suíte de testes consistentes, ele acredita que não conseguirão manter velocidade nas entregas. Por fim, o líder faz um pedido muito claro.

Esse é um bom começo para endereçar o problema. Evidentemente, a desenvolvedora fará suas próprias observações. Quem sabe, ela pode argumentar que não tem tempo para trabalhar nos testes porque os prazos estão muito apertados, ou que simplesmente não sabe como fazê-lo. Com isso, novos pedidos surgirão e um acordo deve ser estabelecido. O acordo, por exemplo, pode ser de trabalharem em par na implementação de testes das próximas três tarefas.

Os quatro componentes da CNV devem ser aplicados nos dois sentidos: expressar-se com honestidade e ouvir com empatia. Enquanto recebemos um *feedback*, é comum que nossa mente imediatamente busque por maneiras de reagir. Antes de entender com profundidade o que está sendo dito, já estamos pensando no que falar para resolver o problema do outro, ou em como podemos nos defender caso nos sintamos atacados. Por isso, quando seguramos nossa reação e tentamos extrair os quatro componentes da CNV em nossa escuta, estamos exercendo uma escuta empática, que proporciona maiores chances de uma resolução positiva do conflito.

É importante alertar que essa maneira de comunicação requer reflexão. Não é natural, porque não fomos treinados para isso. Especialmente no âmbito profissional, isso deve demandar preparação. Se temos um *feedback* para ser dado, seja positivo ou negativo, devemos confrontar nossa opinião com informações prévias para tentar combater vieses e fazer observações e pedidos apropriados. Nas próximas seções, vamos aprofundar cada um dos componentes e entender como eles são aplicados na prática.

Observar sem avaliar

O primeiro componente da CNV é o da observação. Ao fazermos uma observação, buscamos comunicar acontecimentos objetivos, e devemos estar muito atentos para separar uma possível avaliação associada. A avaliação pode existir na comunicação, mas ela deve estar claramente separada da observação. Você poderia avaliar possíveis consequências de um acontecimento sobre você, ou sobre o negócio, mas deixando claro que se trata de sua avaliação dos fatos observados.

Portanto, em uma conversa difícil no trabalho, é importante que você se prepare. Organize os fatos e as evidências. Veja alguns exemplos de observações ruins (com avaliações associadas) e de boas observações (isentas de avaliação):

Observações com avaliações associadas	**Observações isentas de avaliação**
"O projeto está atrasado."	"Já se passou metade do tempo que temos para o projeto e nenhuma de suas partes parecem concluídas no nosso quadro Kanban."
"Sua performance não foi boa no último trimestre."	"Os OKRs da sua equipe ficaram muito aquém do esperado, e não há progresso no seu plano de desenvolvimento individual."

Observações com avaliações associadas	Observações isentas de avaliação
"Você não foi responsável na última liberação."	"Nós trabalhamos com a premissa de '*you build it, you run it*'. Entretanto, quando incidentes aconteceram na semana passada, não houve iniciativa sua e de sua equipe para conter os problemas e mitigar as consequências, antes que outras áreas da empresa fossem afetadas."

Identificando nossos sentimentos

Após a observação, o segundo componente da CNV é sobre identificar nossos sentimentos. Expressar nossos sentimentos plenamente pode ser parte crucial na resolução de conflitos, mas é mais complicado do que parece, porque não estamos acostumados a fazer isso. Na verdade, durante a vida, somos condicionados a suprimir nossos sentimentos ao invés de aprendermos como identificá-los e expressá-los.

Por não sermos bons em falar sobre sentimentos, costumamos confundir sentimentos com julgamentos. Veja os exemplos a seguir:

Exemplos	Comentários
"Eu senti que era inútil pedir ajuda para fulano quando tivemos problemas na semana passada."	Apesar de a frase começar com "eu senti", ela não expressa realmente um sentimento, e sim a maneira como o interlocutor pensa — ele pensou que seria inútil pedir ajuda para fulano. Uma maneira mais apropriada para comunicar o que realmente se passou poderia ser: "quando tivemos problemas na semana

Exemplos	Comentários
	passada, eu me senti desencorajado a pedir ajuda para fulano, porque da última vez ele não estava disposto a ajudar".
"Quando eles marcaram essa reunião, eu me senti ameaçado."	Sentir-se "ameaçado" também não é um sentimento. Essa frase comunica como um fato foi interpretado, mas não o sentimento por trás dessa interpretação. Uma maneira mais apropriada poderia ser: "quando eles marcaram essa reunião, eu senti medo, porque pensei que seria demitido".

Após reconhecermos nosso sentimento, precisamos ser capazes de assumir responsabilidade sobre ele e de comunicar nossas necessidades por trás do sentimento.

Assumindo responsabilidade por nossas necessidades

Por trás de todo sentimento negativo, há uma necessidade não atendida em nós. Mesmo quando estamos com raiva porque julgamos que fomos profundamente ofendidos, ou sofremos injustiça, expressar essa raiva em sua totalidade começa por assumirmos responsabilidade sobre ela. Devemos reconhecer as nossas próprias necessidades não atendidas que proporcionaram esse sentimento.

Julgar as outras pessoas como maus-caracteres, por exemplo, é uma expressão alienada de nossas próprias necessidades insatisfeitas. Antes de procurarmos maneiras de punir as pessoas que julgamos culpadas de nos causarem sofrimento, devemos reconhecer e expressar nossas próprias necessidades — como a necessidade de

respeito, no caso de uma ofensa. Quando fazemos isso, temos mais chances de ver nossas necessidades satisfeitas.

> *"O que os outros fazem podem ser o estímulo para nossos sentimentos, mas não a causa."* – Marshall B. Rosenberg [2]

Veja os exemplos a seguir:

Exemplos	Comentários
"Estou desapontada porque você não me avisou sobre essa vulnerabilidade na aplicação antes."	No exemplo ao lado, a interlocutora atribui a culpa pelo desapontamento no outro, e não reconhece suas necessidades não atendidas por trás desse sentimento. Uma maneira mais apropriada de comunicação poderia ser: "quando você omitiu esta vulnerabilidade, fiquei desapontada porque eu preciso confiar que tenho as informações necessárias para priorizar as próximas atividades da equipe".
"Me sinto injustiçado porque você deu um aumento para fulana."	A palavra "injustiçado" não representa um sentimento. Ela é um julgamento — o interlocutor considera que o outro cometeu uma injustiça. Uma maneira mais apropriada de se expressar poderia ser: "quando soube que fulana recebeu um aumento, eu me senti desencorajado porque eu penso que contribuo da mesma maneira e também gostaria de ser reconhecido".
"Fiquei feliz porque você conseguiu dar conta deste projeto com autonomia."	Assumir responsabilidade sobre nossos sentimentos também é bom ao dar feedbacks positivos. Uma maneira mais apropriada de comunicar este feedback seria: "quando você deu conta deste projeto com autonomia, fiquei feliz porque isso me deu segurança para focar em outras pendências da empresa". Esse é um feedback mais rico, porque justifica a satisfação com mais transparência.

Exemplos	Comentários
"Estou inseguro porque você não me comunica o progresso do projeto."	Da mesma maneira que os exemplos anteriores, o interlocutor não assume responsabilidade sobre os sentimentos causados por sua necessidade de comunicação. O exemplo ao lado poderia ser melhor comunicado da seguinte maneira: "quando você não comunica o progresso do projeto pelos nossos quadros, eu me sinto inseguro porque preciso poder confiar nestas informações para tomar decisões".

Quando reconhecemos sentimentos negativos, devemos evitar apontar culpados — sejam outras pessoas ou nós mesmos. Em vez disso, devemos reconhecer nossas próprias necessidades não atendidas e comunicá-las. Com isso, temos chances maiores de tê-las contempladas. Mas apenas comunicar as necessidades não é suficiente. Precisamos articular um pedido de mudança, o que nos leva ao próximo componente da CNV.

Realizando um pedido

Depois de fazer a observação, identificar o sentimento e comunicar a necessidade por trás dele, precisamos fazer um pedido concreto de mudança. Nesse ponto, o propósito da conversa deve ficar completamente claro. Apenas compartilhar nossa necessidade não é suficiente, porque, para quem está ouvindo, pode não ser óbvia a maneira como a pessoa pode contribuir para o atendimento da necessidade.

Por isso, precisamos formular um pedido claro e objetivo, e remover qualquer ambiguidade que possa ter restado. Veja os exemplos a seguir:

Exemplos	Comentários
"Quando você não comunica o progresso do projeto pelos nossos quadros Kanban, eu me sinto inseguro porque preciso poder confiar nestas informações para tomar decisões".	Nesse exemplo que usamos na seção anterior, uma pessoa está manifestando plenamente suas necessidades. Entretanto, não há um pedido claro para o receptor da mensagem — com que frequência o quadro precisa ser atualizado? Devo adicionar comentários detalhados nas tarefas, ou movê-los pelas colunas é suficiente? Devo fazer as anotações de horas gastas? Por isso, a mensagem deve ser concluída com um pedido claro, que poderia ser: "você poderia atualizar o quadro todos os dias e manter os *cards* nas colunas que representam o estado atual da tarefa?".
"Quando você chegou atrasada à reunião hoje à tarde, fiquei desapontado porque eu gostaria que você tivesse mais consideração pela equipe".	Nesse exemplo, também não há um pedido claro sendo feito. Uma pessoa manifestou o desapontamento sobre o atraso em uma reunião, mas não há um pedido. A pessoa que ouve poderia supor que ela está proibida de se atrasar. Entretanto, o interlocutor poderia apenas desejar ser avisado com antecedência quando isso acontecesse.

Nesse momento, é comum percebermos que não sabemos exatamente o que queremos. Conseguimos expressar plenamente uma necessidade, mas não somos capazes de fazer um pedido. Por isso, a CNV exige reflexão e esforço. Caso você não seja capaz de formular um bom pedido, você não pode esperar uma mudança concreta. Mesmo que em algumas situações desejemos apenas saber o que o ouvinte sente ou pensa sobre o que foi dito, devemos

fazer esse pedido explicitamente para que ele seja incitado a responder.

Por fim, é importante ressaltar que um pedido é apenas um pedido, não uma exigência. A abordagem da CNV é útil para conversas difíceis porque o que queremos como resultado é contemplar as necessidades das pessoas da melhor maneira possível, e isso demanda negociação. Dessa maneira, a CNV não é uma técnica de convencimento — embora você tenha maiores chances de ter suas necessidades atendidas quando comunicadas assim.

Pedidos podem ser negados. Quando isso acontecer, busque também entender os motivos com empatia — extrair as observações, sentimentos e necessidades por trás da recusa. O que determina se o que você disse era um pedido ou uma exigência é o que acontece depois de uma negação.

Ouvir também é parte da Comunicação Não Violenta

Ser capaz de transmitir uma mensagem de maneira eficaz é o primeiro passo. Depois disso, precisamos ouvir com empatia para avaliar a circunstância da maneira mais completa que for possível e chegar a um acordo. Ao ouvir o outro ponto de vista, você tem a oportunidade de reavaliar sua posição e decidir se deseja reafirmá-la ou alterá-la.

Por isso, tenha curiosidade sobre como a outra pessoa vê a situação. Procure extrair os mesmos quatro componentes da CNV antes de pular para respostas e argumentação. Quando estamos em uma conversa difícil, que não é uma situação trivial, precisamos segurar nossos instintos de aconselhar, explicar ou procurar uma solução imediatamente.

Com isso, aumentamos a chance de que a resolução do conflito possua um resíduo positivo — ou seja, que todas as partes saiam compreendidas e com suas necessidades mais bem contempladas do que antes. Por consequência, também aumentamos a chance de

haver melhorias no sistema de trabalho porque as relações se tornam mais saudáveis e resilientes.

Em alguns casos, pode ser importante o envolvimento de outras pessoas para intermediar a resolução do conflito. A função dessas pessoas não é resolver o conflito em si, mas garantir que ambas as partes sejam capazes de expressar suas insatisfações em sua totalidade e, depois disso, colaborar para que os acordos sejam razoáveis para ambas as partes.

6.3 Como dar bons feedbacks?

A abordagem da CNV pode ser uma excelente ferramenta para ampliarmos nossa capacidade de lidar com conversas difíceis que temos no ambiente de trabalho. Porém, há ainda outros aspectos em que precisamos prestar atenção para formular bons feedbacks, sejam eles positivos ou negativos.

Como LeeAnn Renninger, PhD em psicologia cognitiva, comenta no canal TED, no YouTube [3], quando as pessoas tentam dar feedbacks, elas frequentemente caem em dois campos distintos. De um lado, as pessoas comunicam a mensagem de maneira tão suave que a pessoa nem percebe que está recebendo um feedback. De outro lado, a mensagem é comunicada de maneira tão direta que a pessoa sente a necessidade de se defender, o que pode tornar a conversa improdutiva.

LeeAnn Renninger e sua equipe passaram anos entrevistando pessoas boas em dar feedbacks e sintetizaram o que havia em comum em suas abordagens. O resultado da pesquisa foi uma fórmula que contém quatro partes.

A primeira parte é o que eles chamam de **"o micro sim"** (*micro-yes*). Pessoas boas em dar feedbacks começam a conversa com uma pequena e poderosa pergunta. Por exemplo, "você tem alguns

minutos para conversarmos sobre como podemos melhorar a situação com este cliente?", ou "eu poderia compartilhar com você minhas ideias sobre como podemos evitar que o incidente de ontem não se repita?".

Essa pergunta inicial possui duas funções importantes. Primeiro, ela deixa claro que um feedback está para ser dado, e deixa a pessoa atenta para a conversa. Em segundo lugar, a pergunta tem o papel de adquirir um comprometimento inicial, de forma a demonstrar que a pessoa está disposta a falar sobre um assunto.

A segunda parte da fórmula se chama **"apontamento de dados"**. Ela é similar à etapa de observação da CNV. Trata-se de trazer os fatos e informações específicas sobre a pauta da conversa. Nesse momento, queremos evitar avaliações precipitadas e evitar misturar julgamentos com observações.

A terceira parte se chama **"apresentar o impacto"**. Esta também é similar à parte de necessidades da CNV. Queremos deixar claro quais foram as consequências, positivas ou negativas, do que foi observado. Isso dá a noção de propósito para os fatos trazidos anteriormente e revela por que a conversa é relevante.

Por fim, a quarta parte é uma pergunta. Pessoas boas em dar feedbacks terminam seus pensamentos com uma **pergunta que convida à participação**. Por exemplo, "essas são as minhas ideias, mas como você vê isso?", ou "isso faz sentido para você?". Este é um convite sincero para que a pessoa participe da construção de uma mudança e, também, para que se obtenha o seu comprometimento, em vez de complacência.

Apesar de todas as dicas mencionadas ao longo do capítulo até agora, é importante deixar claro que dar feedbacks negativos sempre causará sofrimento. Uma conversa de demissão, por exemplo, jamais será fácil. Apesar disso, quanto antes um feedback

negativo for comunicado, menor será esse sofrimento — seja para o indivíduo ou para o grupo de que a pessoa faz parte.

Entretanto, feedbacks são necessários para o crescimento das pessoas e da organização, porque são apenas através deles que as organizações podem se transformar e evoluir.

As abordagens mencionadas no capítulo devem servir como ferramentas para que estas interações possam acontecer da maneira mais produtiva possível. Elas não são maneiras de manipular as pessoas para convencê-las a fazer o que queremos. Quando usadas dessa maneira, elas não serão eficazes.

Por fim, muitas das conversas difíceis que teremos no ambiente de trabalho terão a ver com falhas. As técnicas mencionadas neste capítulo devem ajudar, mas devemos aprofundar a maneira como lidamos com falhas no ambiente de trabalho. Esse será o tema do próximo capítulo.

Resumo

- Conversas são difíceis no trabalho porque temos nossa identidade bastante conectada com nossa profissão. Isso leva a emoções fortes, ao mesmo tempo em que não somos capacitados para falar no trabalho sobre nossos sentimentos e necessidades — não sabemos como fazer isso;
- A Comunicação Não Violenta (CNV) propõe uma estrutura de comunicação com quatro componentes fundamentais: observação, sentimento, necessidade e pedido. Essa estrutura facilita uma comunicação plena e é útil na resolução de conflitos e conversas difíceis;
- Bons feedbacks não podem ser suaves demais, ou diretos demais. Uma boa resolução de um conflito não é quando a pessoa é convencida a fazer o que queremos, mas quando ambos se comprometem a contemplar as necessidades não atendidas.

Referências

[1] ROSENBERG, Marshall. *Comunicação não-violenta*: técnicas para aprimorar relacionamentos pessoais e profissionais. São Paulo: Ágora, 2006.
[2] *Ibidem*, p. 79.
[3] RENNINGER, LeeAnn. (TED). *The secret to giving great feedback | The Way We Work, a TED series*. YouTube, 10 fev 2020. Disponível em: https://www.youtube.com/watch?v=wtl5UrrgU8c. Acesso em 16 set. 2022.

Capítulo 7
Lidando com falhas

Sabemos que precisamos aprender com erros, mas é mais fácil falar do que fazer. Na nossa carreira, teremos que conviver com falhas de muitas naturezas. Falhas serão cometidas ao entregar software que não faz o que deveria fazer ou com bugs; ao planejar projetos e realizar alocação de pessoas. Falhas serão cometidas do ponto de vista estratégico, quando a empresa faz algo que não deveria ser feito para caminhar em direção aos seus objetivos de negócio. Falhas serão cometidas nos relacionamentos interpessoais, na gestão de conflitos e na incapacidade das pessoas de exporem com clareza seus sentimentos e necessidades. Falhas serão cometidas ao se lidar com falhas.

Falhas vão acontecer porque, como falamos em capítulos anteriores, não podemos desenvolver planos perfeitos com antecedência. Descobertas precisarão acontecer enquanto o trabalho acontece. É apenas a partir dos feedbacks que conseguimos ajustar a direção do desenvolvimento de produtos.

Mesmo sabendo que precisamos aprender com nossos erros para evoluirmos como profissionais e empresas, falhas sempre causarão algum tipo de sofrimento, como culpa, decepção ou frustração. Apesar de conscientemente reconhecermos que isso faz parte da natureza do trabalho, erros vão continuar nos causando sentimentos negativos porque eles trarão consequências, sejam individuais ou para o grupo.

Portanto, devemos estar atentos à maneira como interpretamos e agimos em relação aos diferentes tipos de falhas que fazem parte do nosso dia a dia. Já que isso é inevitavelmente parte da nossa profissão, devemos aprender a lidar com erros e buscar a sustentabilidade humana do trabalho e a redução do sofrimento.

Apesar de não sermos capazes de evitar falhas completamente, nem tudo está jogado ao acaso. Somos capazes de nos preparar, em alguma medida, ao aprender com metodologias e com a experiência de outras pessoas. Neste capítulo, vamos aprofundar os diferentes tipos de falhas que acontecem em nossas organizações, para pensarmos em maneiras de agir que vão além da investigação de causa e efeito.

7.1 Um olhar sistêmico para falhas

Quando uma falha acontece, podemos abordá-la ao menos de duas maneiras diferentes. Primeiro, vamos analisar o incidente na perspectiva de causa e efeito. Por exemplo, quando um bug surge, investigamos a ação exata que o originou. Buscamos qual foi a mudança exata no código-fonte que causou o comportamento indesejado do sistema e quem a implementou. Com isso, investigamos os motivos das alterações e resolvemos o problema com uma correção. Essa é a dinâmica do pensamento linear, de causa e efeito.

Uma segunda maneira de abordar o erro é pensar no sistema de trabalho como gerador de bugs. Nesse caso, a busca é pela maneira como podemos reconfigurar nosso esquema de trabalho para reduzir as possibilidades de surgimento de bugs. Você não resolve os problemas definitivamente, mas cria mecanismos, configurações e acordos para reduzir a chance de eles aparecerem.

Ambas as abordagens são importantes. O entendimento de causa e efeito pode ser o primeiro passo na investigação de um incidente. Mas, depois disso, as ações devem ser ampliadas para um ponto de vista mais sistêmico. Devemos observar as circunstâncias de uma falha para implementar melhorias mais abrangentes do que o incidente específico. Algumas destas melhorias serão técnicas — como a implementação de um *pipeline* de *deploy* automatizado e

uma suíte de testes — e outras serão culturais — como a adoção de uma cultura DevOps ou a mudança como uma organização lida com falhas.

As mudanças culturais são as que possuem maior potencial transformador. Quando uma organização constrói uma cultura que é positiva para o negócio, as atitudes escalam por toda a companhia, e não se restringem a uma iniciativa localizada. No caso da adoção de uma cultura DevOps, por exemplo, as equipes vão se mover para criar e manter o ferramental necessário que possibilite serem responsáveis pelo software em produção de maneira mais autônoma e segura que for possível. Mais importante do que as ferramentas em si, as equipes vão entender o sentido de elas existirem.

Portanto, muitas vezes, a ocorrência de uma falha deve proporcionar uma avaliação mais profunda sobre como a empresa trabalha e o que ela valoriza. Quem sabe a resposta para um bug que permanece por uma semana no ambiente de produção não seja apenas resolvê-lo, ou criar mecanismos de alerta (embora alertas sejam fundamentais), mas também gerar uma reflexão mais profunda sobre a responsabilidade do trabalho entregue pelas equipes.

Quando a investigação de uma falha se resume à análise de causa e efeito, as ações subsequentes se restringem a uma: buscar culpados. Com essa maneira de pensar, quando houver um incidente de indisponibilidade de uma aplicação, um programador deve ser culpado pelo descuido. Isso proporciona uma cultura tóxica, porque gera os incentivos por qualidade através do medo. Ao invés de resultar em mais qualidade, as pessoas e equipes são estimuladas a esconderem seus erros, o que acumula riscos maiores para a organização.

A prática de escrever um documento de *Post Mortem* após um incidente grave — que envolve, por exemplo, *downtime* ou perda de dados — é uma excelente maneira de compartilhar o aprendizado adquirido dentro da organização de uma maneira construtiva, quando feito corretamente. O *Post Mortem* consiste em um

documento que descreve com detalhes o incidente, suas causas e possíveis ações preventivas para reduzir a recorrência ou impacto de incidentes semelhantes no futuro. Mas, para que este documento não seja interpretado como uma punição, ele não deve ser apontar culpados. A investigação deve partir da premissa de que todos tinham a intenção de fazer o melhor com as informações que tinham no momento.

> *"Você não pode 'consertar' pessoas, mas você pode consertar sistemas e processos para ajudar pessoas a fazerem as escolhas certas ao projetarem e manterem sistemas complexos."* – John Lunney e Sue Lueder [1]

7.2 A influência do negócio sobre como lidamos com falhas

A análise de uma falha, de sua causa e efeito, assim como o mapeamento de oportunidades de melhorias no sistema de trabalho no âmbito cultural e técnico, não pode ignorar os sistemas maiores dos quais o software faz parte ou com os quais interage — isto é, geralmente, o negócio que dá sentido à existência do software.

Negócios diferentes podem possuir necessidades muito distintas em termos de escalabilidade, confiabilidade ou manutenibilidade. Alguns softwares requerem uma capacidade alta de manter disponibilidade frente a um grande volume de requisições, como um portal de notícias, por exemplo. Nesse caso, se um artigo viralizar, o portal deve ser capaz de escalar e suportar as requisições — caso contrário, se a aplicação ficar fora do ar, os usuários buscarão as informações no site concorrente. A confiabilidade não precisa ser das melhores, não é uma catástrofe se uma requisição ou outra for perdida, mas a escalabilidade é crítica.

Por outro lado, no sistema interno de um hospital, por exemplo, a escalabilidade não é crítica — não se espera que haja uma mudança brusca de carga computacional nesse contexto. Entretanto, a confiabilidade é muito importante. Médicas e enfermeiros devem confiar na precisão e disponibilidade dos dados para tomar decisões que influenciam diretamente a saúde dos pacientes.

Nesses dois exemplos, um mesmo erro técnico pode ter consequências completamente diferentes. Imagine que há o problema de *N+1 queries* em ambas as aplicações — quando um código executa *N+1* vezes uma operação no banco de dados que poderia ser feito em apenas uma. Esse mesmo erro pode ser insignificante para o sistema hospitalar, que recebe poucas requisições, mas poderia ser crítico para o portal de notícias, porque pode degradar consideravelmente a capacidade de a aplicação escalar.

Portanto, o negócio em que o software está inserido influencia diretamente a maneira como lidamos com falhas. Um mesmo bug pode significar coisas diferentes para diferentes empresas, e, sendo assim, não podemos ter uma receita de como agir para cada um deles. O mesmo pode acontecer em fases diferentes de uma mesma empresa. Para uma startup que está começando, por exemplo, a manutenibilidade pode ser não relevante se 80% do produto é descartado a cada 3 meses. Entretanto, quando a empresa se aproximar do *product-market fit*, a manutenibilidade definitivamente será importante para que o produto possa continuar evoluindo rápido e com equipes maiores.

7.3 Três tipos de falhas

Mesmo com tantas variáveis que influenciam a maneira de agir na ocorrência de erros, precisamos falar de uma maneira mais prática. Para isso, vamos aprofundar nos três tipos de falhas que Amy Edmondson descreve em seu livro *The Fearless Organization* [2], que já foi mencionado em capítulos anteriores. Estes três tipos de falhas são: falhas evitáveis, falhas complexas e falhas inteligentes.

Falhas evitáveis

Falhas evitáveis são decorrentes do não cumprimento de um procedimento já conhecido que evitaria a falha. Portanto, são falhas previsíveis. A empresa já possui, ou conhece, algum mecanismo para evitar a falha, mas, por algum motivo, ele não funcionou ou não foi implementado. Por exemplo:

- Uma release foi liberada, mas foram ignorados testes que quebraram no CI. Isso levou bugs ao ambiente de produção;
- Boas práticas não foram aplicadas na implementação de um algoritmo, e agora o código está difícil de manter. Por consequência, a equipe não consegue entregar melhorias em tempo hábil;
- Após a liberação de um novo *dashboard*, *timeouts* em requisições começaram a acontecer devido a uma *query* pesada em um banco de dados relacional que não possuía índices apropriados para respondê-la.

Sabemos que os exemplos anteriores podem acontecer em projetos de software. Também conhecemos maneiras de evitá-los de maneira satisfatória. Por exemplo, podemos ter processos mais burocráticos no *pipeline* de liberação que impedem que uma release seja liberada sem que todos os testes passem. Podemos estabelecer um processo de revisão de código, que exige a aprovação de um desenvolvedor sênior em todos os *pull requests*, para garantir um certo nível de manutenibilidade do código. Por fim, podemos implementar testes de performance para funcionalidades que oferecem riscos maiores de conter problemas de lentidão.

Essas ações não impedem de maneira definitiva a ocorrência dos incidentes, mas reduzem a chance de eles acontecerem. Assim, o que se busca é evoluir o sistema de trabalho para que ele evite estas falhas de maneira satisfatória — e o que isso significa dependerá do contexto do negócio.

Como já falamos, uma pequena startup em busca de *product-market fit* possui necessidades bem distintas de uma organização consolidada, com centenas de engenheiros e engenheiras de software e milhares de clientes. Assim, manter uma infraestrutura robusta e sofisticada em termos de escalabilidade, por exemplo, não fará sentido frente ao verdadeiro risco da companhia, que é não encontrar um modelo de negócio sustentável.

Essa diferença na forma de agir frente a uma mesma falha, em contextos diferentes, se justifica porque a probabilidade de elas acontecerem também são diferentes, assim como as consequências e dificuldade em mitigá-las. No exemplo anterior, investigar um erro de lógica na startup — talvez com seus fundadores sendo autores de todo o código produzido, com acesso a todos os logs e dados do ambiente de produção — pode ser uma tarefa muito mais simples do que na empresa consolidada, que possui mais equipes e aplicações interdependentes.

Mesmo quando implementamos procedimentos e automações para melhoria do sistema de trabalho, elas ainda podem ser burladas ou conter falhas. Por isso, mais importante do que a ferramenta em si, uma mudança cultural deve estar associada a ela. O objetivo ao implementar testes automatizados não é ter cobertura de 100% do código, mas oferecer um produto com qualidade em que os usuários possam confiar. Da mesma maneira, monitoramentos sofisticados da aplicação não vão garantir mais disponibilidade do produto por si só se ninguém se sentir responsável pelos alertas e métricas — o objetivo não é a implementação da ferramenta, mas a mudança de postura.

Motivações contraditórias

Além dos desafios de elaboração de procedimentos, implementação de ferramentas e mudanças culturais, falhas evitáveis também têm relação com os incentivos contraditórios a que pessoas e empresas são submetidas. Por exemplo, se eu sou um desenvolvedor sênior em uma empresa e sou chamado para uma reunião importante com a diretoria para discutir uma proposta comercial que precisa ser enviada no mesmo dia para um potencial cliente, e, ao mesmo tempo, um alerta de erro de um produto da empresa chega por e-mail, qual é a decisão certa a ser tomada: devo primeiro investigar o incidente ou ir à reunião?

Pode ser que o alerta seja um falso-positivo, ou que a reunião com a diretoria não leve a nada. Posso pensar que, ao mesmo tempo em que preciso ser responsável com o alerta que chegou, também devo ser responsável com a proposta que precisa ser enviada no mesmo dia, pois isso poderá ter consequências grandes para a companhia. Então, o que fazer? Não há resposta certa *a priori* que poderia ser dada; apenas a pessoa inserida no contexto poderia ter informações suficientes para apostar em uma decisão naquele momento.

Dessa maneira, falhas evitáveis não estão apenas relacionadas a procedimentos e ferramentas, embora elas sejam fundamentais para evitá-las. Também não se trata apenas de disciplina ou burocracia. Elas também estão relacionadas com a clareza que as pessoas têm sobre os objetivos e valores da empresa. Elementos como estes, que são mais subjetivos, precisam orientar as pessoas a fazerem avaliações em momentos em que há motivações contraditórias.

Apesar de olharmos para o sistema de trabalho como gerador de falhas, as pessoas não podem se eximir da responsabilidade. Elas devem ser capazes de reconhecer os riscos que estão assumindo e garantir que estes estão alinhados com os interesses da companhia.

Além disso, o sistema de trabalho sempre contará com a boa vontade, caráter e responsabilidade de seus participantes — seja para mapear riscos, comunicá-los ou propor melhorias. Especialmente, tratando-se de falhas que envolvem má-fé, os

indivíduos devem ser responsabilizados e punições podem acontecer. Se uma organização descobre que o RH da empresa circulou dados sensíveis de colaboradores de maneira intencional, evidentemente a pessoa deve ser punida pelos seus atos.

Uma melhoria na gestão de dados — que garanta que os colaboradores têm apenas o privilégio mínimo necessário de acesso a dados para desempenhar suas funções profissionais — deve ser executada também, mas isso não exime a responsabilidade individual de quem cometeu o ato.

Falhas complexas

Diferente das falhas evitáveis, quando já conhecemos procedimentos que podem evitá-las, falhas complexas são difíceis de prever. O ambiente em que um software está inserido é caótico e não somos capazes de visualizar todas as suas possibilidades. Por consequência, não somos capazes de prever e nos preparar para todas as situações possíveis.

Imagine uma situação em que houve um incidente de *downtime* em uma aplicação por 2 horas, o que levou a prejuízos financeiros para uma empresa. O incidente aconteceu porque uma nova *release*, que continha uma funcionalidade nova para usuários *premium*, foi liberada, ao mesmo tempo em que uma ação de marketing da companhia viralizou e, de um dia para o outro, o crescimento esperado de usuários para o semestre inteiro aconteceu.

Olhando para trás, é provável que as pessoas sintam que poderiam ter previsto e evitado o incidente. Elas podem olhar para trás e reconhecer que a nova funcionalidade tinha uma vulnerabilidade em termos de escalabilidade que poderia ter sido mitigada antes da liberação. Dessa maneira, como falamos anteriormente, a equipe poderia evoluir seu sistema de trabalho e criar mecanismos que poderiam de fato evitar problemas similares no futuro — pelo menos uma parte significativa deles. Entretanto, há dois motivos principais que impedem que falhas complexas sejam sempre evitadas.

Primeiro, criar mecanismos para evitar falhas complexas é caro. Você precisa de um esforço grande das pessoas que conhecem profundamente o software para implementá-los e mantê-los funcionando enquanto o produto evolui. Nem sempre esse investimento vale a pena.

O segundo motivo é que o conhecimento necessário para implementar esses mecanismos emerge ao observar o sistema em ação — ou seja, ao observar o software ser utilizado para resolver um problema no mundo real, e não hipotético. Evidentemente, gerentes de produto e desenvolvedoras experientes podem ter um olhar mais abrangente para visualizar riscos. Mesmo assim, não podemos esperar que podemos nos desenvolver como profissionais ao ponto de sermos capazes de prever e mitigar todos os problemas de uma aplicação antes de acontecerem.

Portanto, apesar de falhas complexas serem diferentes das falhas evitáveis, elas têm uma maneira similar de agir. As equipes devem investigar os problemas, mas não para procurar culpados. A motivação é evoluir o sistema de trabalho para que ele seja mais robusto em relação a falhas no futuro.

Falhas inteligentes

Por fim, temos as falhas inteligentes. Apesar de os dois tipos de falhas anteriores gerarem aprendizado e proporcionarem o crescimento da equipe, não podemos comemorá-las. São falhas que, apesar de tudo, são lamentadas — seria melhor que elas não tivessem acontecido. Nesse sentido, as falhas inteligentes são diferentes. São falhas que, quando reconhecidas, podem economizar tempo e dinheiro da empresa. Apesar disso, nem sempre são fáceis de serem comunicadas.

Imagine que uma companhia esteja desenvolvendo um novo produto. A equipe de UX e produto investiu meses se debruçando sobre processos e investigando a necessidade dos potenciais clientes. Eles têm uma boa ideia do que fazer para resolver os

problemas — conhecem os casos de uso e histórias de usuários. Com uma proposta de solução bem delimitada, a equipe de desenvolvimento começa a se envolver no planejamento de um MVP (mínimo produto viável) da solução, e estima-se que precisarão usar os próximos 3 meses para a implementação. O tempo de implementação está dentro das restrições do negócio — mesmo com um orçamento limitado, a empresa pode desenvolver o produto, e eles seguem com o plano.

Entretanto, na segunda semana de desenvolvimento, o gerente de produto percebe que há um problema regulatório na solução. Ninguém percebeu que a solução infringia leis de privacidade e proteção de dados do país em que ela seria utilizada e, por conta disso, o aplicativo não poderia ser publicado da maneira em que foi planejado.

Evidentemente, houve uma falha. Durante a exploração de um novo terreno para o desenvolvimento de um produto, a companhia cometeu um erro em seu planejamento, deixando passar aspectos fundamentais que precisavam ser considerados no início. Todavia, o reconhecimento dessa falha nessas circunstâncias deve ser comemorado. A equipe estaria em uma situação muito mais difícil depois de três meses se isso não tivesse acontecido. Com essa descoberta, eles terão o desafio de rever suas premissas e redefinir a solução em um período mais curto, o que é desafiador, mas certamente mais favorável.

Pode parecer óbvio que, nestas condições, o gerente comunicaria sua descoberta sem hesitar. Entretanto, em um contexto em que há incentivos contraditórios, isso pode não ser uma decisão tão clara. Imagine que essa pessoa deseje uma promoção e que uma *performance review* estaria para acontecer em um mês. Para a sua *performance review*, poderia ser mais interessante para ela não comunicar uma notícia ruim dessa magnitude e, no futuro, tentar se eximir da responsabilidade.

O que vai influenciar a maneira como as pessoas vão agir nestas situações é como a cultura da empresa lida com falhas. Quando a cultura da empresa lida com falhas usando o pensamento linear, de causa e efeito, procurando por culpados, as pessoas tendem a se comportar defensivamente. Por outro lado, quando há um entendimento mais sistêmico da organização, e a maneira como a organização lida com falhas fortalece um alinhamento de propósito individual e coletivo, falhas inteligentes podem emergir.

Nesta segunda parte do livro, falamos sobre vários aspectos da cultura das empresas em que estamos inseridos. Redefinir o que nossas organizações valorizam, aprender a ter conversas difíceis e ressignificar a maneira como interpretamos falhas são tópicos fundamentais para a construção de um ambiente de trabalho humanamente mais sustentável.

Ainda assim, para falarmos sobre saúde mental e sustentabilidade humana, precisamos entrar no campo individual. É bem possível que, mesmo em uma companhia com boas práticas de gestão e uma boa cultura, podemos nos colocar em condição de *burnout* por causa de padrões de pensamentos e crenças individuais. Falaremos sobre isso na próxima parte.

Resumo

- Sabemos que precisamos aprender com falhas, mas é mais fácil falar do que fazer. Precisamos mudar nossa maneira de agir, comumente orientada pelo pensamento linear, onde buscamos os culpados pelas falhas, para um pensamento sistêmico, onde interpretamos o sistema de trabalho como produtor de falhas, e buscamos configurações que reduzem a chance de elas acontecerem;
- O sistema de trabalho não evolui apenas com a implementação de procedimentos e mecanismos. Uma mudança mais significativa, potencialmente mais transformadora, se dá no nível cultural;
- Falhas evitáveis são decorrentes do não cumprimento de um procedimento já conhecido que poderia evitá-la. Falhas complexas são mais difíceis de serem previstas, e acontecem no ambiente caótico de produção. Falhas inteligentes são falhas que precisam ser celebradas, porque, quando reconhecidas, podem evitar prejuízos maiores para a companhia.

Referências

[1] LUNNEY, John; LUEDER, Sue. Postmortem Culture: Learning from Failure. *In*: GOOGLE. *Site Reliability Engineering*. (Website). O'Reilly Media, Inc., 2017. Disponível em: https://sre.google/sre-book/postmortem-culture/. Acesso em 16 set. 2022.

[2] EDMONDSON, Amy C. *The Fearless Organization: Creating Psychological Safety in the Workplace for Learning, Innovation and Growth*. John Wiley & Sons Inc., 2018.

PARTE 3 - O INDIVÍDUO E SUAS CRENÇAS

Capítulo 8
Padrões de pensamentos

Na primeira e segunda parte deste livro, falamos sobre vários aspectos organizacionais que influenciam a sustentabilidade do trabalho. Esses aspectos estavam relacionados a como as equipes trabalham e ao que a cultura das empresas valoriza. Mas, apesar de essas características influenciarem muito a nossa saúde mental, também podemos cuidar de nós mesmos — podemos ser agentes de transformação e influenciar, de maneira positiva ou negativa, a nossa própria relação com o trabalho.

Quando nossas crenças individuais mais profundas são negativas, um ambiente organizacional saudável não será suficiente para a manutenção de uma boa saúde mental. Nesse caso, precisamos investigar nossos próprios pensamentos e desconstruir distorções cognitivas que possam existir. Investigar nossos diálogos internos é o tema deste capítulo.

O primeiro passo para começar esta investigação é entender que o que pensamos não é realidade absoluta. Com frequência, interpretamos um determinado fato atribuindo ideias falsas em volta dele. Se um colega de trabalho de repente age de maneira mais ríspida conosco, por exemplo, atribuímos uma causa de maneira arbitrária — podemos pensar que ele deve estar chateado por algo que fizemos.

Em outros momentos, fazemos julgamentos sobre nós mesmos também desconectados da realidade. Por exemplo, penso que não mereço liderar um novo projeto da empresa, quando, na verdade, não há outra pessoa mais preparada para estar na posição.

Por isso, não podemos tomar como verdade todos os pensamentos que nossa mente produz. Devemos investigá-los e separar o que realmente condiz com o que podemos observar da realidade. Para começarmos, aprofundaremos os padrões de pensamentos que podem nos levar à Síndrome de *burnout*.

8.1 Síndrome de burnout

O *burnout* é uma síndrome psicológica que emerge de uma prolongada condição de estresse crônico no trabalho [1]. Ele está relacionado a um estado de exaustão e pode ser observado pela manifestação de sintomas como isolamento, ansiedade, depressão e baixa autoestima. Manifestações físicas também são comuns, como insônia, dores musculares, cansaço constante e distúrbios gastrointestinais.

Apesar de o *burnout* estar sempre associado à condição de exaustão, há outras duas dimensões que devem ser consideradas. Uma delas é o cinismo, que engloba a dificuldade do indivíduo em relacionar-se com outras pessoas e engajar-se em atividades do trabalho; a outra é o senso de ineficácia, que captura a autopercepção da pessoa sobre o valor do seu trabalho e a qualidade de sua contribuição.

O *burnout* também tem uma relação complexa com a saúde física. Uma saúde precária contribui para o *burnout*, e o *burnout* contribui para uma saúde precária. Como demonstrado em um estudo longitudinal de 10 anos, a prevalência de *burnout* possibilitou prever a subsequente internação em hospitais por problemas cardiovasculares de uma população [2].

Apesar de ainda haver muito debate sobre qual a melhor maneira de medir *burnout*, e quais são suas dimensões, o *burnout* passou a ser reconhecido como fenômeno ocupacional pela OMS em 2019 [3]. Com

essa interpretação, empresas poderiam ser responsabilizadas pelas consequências severas na saúde mental e física de seus trabalhadores causadas pelo estresse crônico do ambiente de trabalho.

Em uma apresentação no evento *DevOps Enterprise Summit* em 2018, a Dra. Christina Maslach, uma das pesquisadoras mais relevantes na área de saúde ocupacional, argumenta que há seis caminhos para a manutenção de um ambiente de trabalho saudável [4]:

- Carga de trabalho sustentável — há momentos para recuperação na rotina de trabalho?
- Capacidade de escolha e controle — as pessoas têm autonomia para decidir como o trabalho deve ser feito?
- Reconhecimento e recompensa — há reconhecimento sobre o bom desempenho dos trabalhadores?
- Comunidade de trabalho apoiadora — há confiança suficiente entre as relações de modo a facilitar a resolução de conflitos?
- Justiça e respeito — as políticas e valores da empresa são colocados em prática de maneira justa e respeitosa?

Apesar de os estudos na área de estresse ocupacional e *burnout* se voltarem às características do ambiente de trabalho, e que, como vimos nas partes anteriores deste livro, sabemos que elas influenciam fortemente a sustentabilidade do trabalho, devemos também considerar a perspectiva individual. Para isso, aprofundaremos sobre o conceito de autoafirmação e autoexploração do filósofo sul-coreano Byung-Chul Han.

8.2 A dinâmica da autoafirmação e autoexploração

Como Byung-Chul Han argumenta em seu livro *Sociedade do Cansaço* [5] a condição de *burnout* pode surgir de uma dinâmica de autoexploração. Mesmo em um ambiente que preza por uma carga de trabalho sustentável, há crenças mais profundas no espírito do nosso tempo que podem nos levar ao esgotamento.

Na economia do conhecimento e na forma como as relações de trabalho tendem a se estabelecer na atualidade, a pressão nem sempre é externa. Nós, como indivíduos, colocamos pressão sobre nós mesmos pela dinâmica da autoafirmação.

A autoafirmação acontece por meio de projeções que fazemos sobre nós mesmos. Nós construímos uma idealização futura de nós (o eu-ideal), e com isso criamos uma discrepância em relação à nossa condição atual (o eu-real). Por exemplo, quando almejamos atingir certos objetivos profissionais, como conseguir uma promoção, criar um novo negócio ou conseguir um emprego específico, estamos nos projetando no futuro. A partir disso, também somos individualmente responsáveis por resolver essa discrepância criada.

Dessa maneira, a autoafirmação não é necessariamente ruim. É por meio desse processo que nos movemos para nos construirmos como pessoas e que somos instigados a atribuir sentido ao que fazemos. Entretanto, o que acontece quando a autoafirmação não se concretiza? O que acontece quando não conseguimos resolver a discrepância entre o eu-real e o eu-ideal? Nesse caso, é possível que haja um autodesprezo — como responsáveis por nosso próprio fracasso, sofremos por uma falta de autoaceitação e nutrimos uma espécie de ódio por nós mesmos.

Com o excesso de positividade do nosso tempo, que diz que tudo que desejamos depende exclusivamente de nós e de nosso esforço, iniciamos um ciclo de autoexploração. Dado que tudo depende de mim, qual é o limite do meu esforço para eu me tornar quem eu desejo?

> *"A autoexploração é muito mais eficiente que a exploração estranha, pois caminha de mãos dadas com o sentimento de liberdade. Paradoxalmente, o primeiro sintoma do _burnout* é a euforia. Lançamo-nos eufóricos ao trabalho. Por fim, acabamos quebrando."_ – Byung-Chul Han [6]

Assim, na dinâmica da autoafirmação não há um vilão em comum, algo com o que as pessoas podem se revoltar. O sofrimento dos indivíduos não se junta em uma causa coletiva porque a exploração não se dá por meio do controle de uma pessoa sobre outras. Tratam-se de problemas que as pessoas devem resolver por elas mesmas. No máximo, podem inspirar-se em outros exemplos, como influenciadores, mas estas pessoas não possuem nenhuma responsabilidade sobre as consequências do que falam — e as consequências podem ser as piores em uma cultura que valoriza realizações a qualquer custo.

Trabalhamos incessantemente não apenas para nos encaixarmos no mercado de trabalho, mas também para satisfazer nossa autoafirmação. É comum tratarmos nosso próprio conhecimento como produto necessário para essa autoafirmação — por exemplo, quando acreditamos que precisamos aprender sobre o novo *framework* lançado para podermos nos afirmar como desenvolvedores modernos.

Ao mesmo tempo, essa projeção que fazemos sobre nós mesmos é o que nos impulsa a nos desenvolvermos como pessoas. Quando não exercemos essa projeção, eliminando todo o risco e o medo do fracasso, também eliminamos tudo que pode nos fortalecer e nos inspirar. O sofrimento é, portanto, um ingrediente necessário para a vida. Não devemos ir em busca do sofrimento, mas também não devemos evitá-lo a qualquer custo, porque não podemos experimentar contentamento sem arriscar experimentar a dor.

Investigando nossas motivações

Sem podermos escapar da dinâmica da autoafirmação, precisamos aprender a lidar com ela para não cairmos em ciclos de autoagressão. Precisamos ter consciência do que se passa em nossa mente, porque, caso contrário, acabaremos por deixar que o acaso determine o que fazemos e como nos sentimos. Embora não possamos ter controle sobre tudo o que acontece em nossas vidas, queremos ter consciência e atribuir sentido à direção que estamos tomando.

Nessa autoinvestigação de nossas motivações, é comum percebermos que agimos para nos colocar em conformidade com as normas sociais e obter reconhecimento. No desenvolvimento de software, por exemplo, isso pode significar buscarmos conformidade com os caminhos adotados por uma comunidade em termos de metodologias e tecnologias.

Porém, em outros casos, em vez de agirmos de acordo com o que esperam de nós, nós confiamos em nosso instinto e construímos uma certa autenticidade na maneira de agir. Nesse caso, estamos exercendo um salto de fé. Quando pessoas trocam de emprego para ter um salário menor, por exemplo, elas estão exercendo um salto de fé — elas confiam que, apesar de o resultado financeiro ser menor no curto prazo, a decisão se justificará no futuro porque estará alinhada com quem a pessoa deseja se tornar.

Relacionamentos também são bons exemplos de saltos de fé. Um vínculo duradouro só pode ser estabelecido por um salto de fé porque nada pode garantir que ele dará certo *a priori*, ao mesmo tempo em que ele só tem chance de dar certo se ambas as pessoas acreditarem que dará no início.

Não queremos escolher entre uma ou outra dessas duas maneiras de agir, que de um lado busca conformidade com o mundo externo e, de outro, autenticidade. O que buscamos é criar uma harmonia entre elas — devemos confiar em nossos saltos de fé, ao mesmo tempo em que temos uma sensibilidade com o mundo externo com que nos relacionamos.

Por isso, sugestões como "faça o que você ama", ou "faça o que tiver vontade", que são carregadas por um excesso de positividade, não são bons conselhos. Eles ignoram o fato de que precisamos nos encaixar no mundo e que boa parte de nossa satisfação vem da nossa relação com o mundo externo. Essas interações podem se dar, por exemplo, por meio do reconhecimento que obtemos das pessoas que consideramos, ou pela nossa condição financeira.

Ao mesmo tempo, focar apenas nos resultados extrínsecos também não é um bom caminho. Nesse caso, colocaremos boa parte de nossa motivação ao acaso, por dependerem de aspectos que muitas vezes estão fora de nosso controle. Por isso, o que buscamos é uma reconciliação entre essas duas formas de motivações — nossos saltos de fé devem ser levados em conta, assim como a maneira pragmática que agimos para nos encaixarmos no mundo e transformá-lo.

Lidando com a ansiedade e frustração

Além de investigarmos nossas motivações por trás da autoafirmação, também precisamos lidar com dois aspectos que são conectados: ansiedade e frustração. Quando experimentamos um domínio da ansiedade sobre nós, nossa mente se volta completamente para uma situação futura hipotética. Por exemplo, quando ficamos apreensivos antes de uma apresentação, imaginando uma reação negativa da plateia.

Em outros casos, a ansiedade surge de uma expectativa positiva, como quando ansiamos pela entrega de um produto e suas consequências — como clientes satisfeitos, reconhecimento profissional ou resultados financeiros. Quando geramos expectativa sobre uma situação futura, eventualmente acabaremos frustrados — como quando o que idealizamos não se concretiza, ou quando não gera os resultados que esperávamos.

Essas situações, assim como várias outras que apresentamos neste livro, são problemas perenes. Não podemos evitar que aconteçam,

mas podemos aprender a lidar com elas de maneira a reduzir o sofrimento. Para o problema da ansiedade, uma possível abordagem é focar na situação presente:

> *"O convite da liderança é para amar a situação presente — como é. A oportunidade existe no presente: resolver o problema; e não no futuro: na entrega final e derradeira do produto. O que há no futuro é a afirmação de uma potencialidade. Entre o presente e o futuro está sua vontade. Engajar significa construir uma vontade de realizar um potencial enquanto se está envolvido por uma espécie de amor pelo que se tem, pelo que se é."* – Alisson Vale [7]

Em nossos projetos, é comum pensarmos que, quando alcançarmos determinado *milestone*, finalmente estaremos satisfeitos. Sabemos que este momento nunca chega, e que, à medida que nos aproximarmos do que almejamos, já estaremos definindo nossas próximas metas. Entretanto, o problema não está no estabelecimento de metas e no desejo de mudança, mas na relação de desprezo que temos com a situação presente.

Quando nos sentimos frustrados por algo, não é o fato em si que é relevante, mas a maneira como o interpretamos. Esse é o princípio básico da Terapia Cognitivo-Comportamental. Pense, por exemplo, no caso de uma pessoa que foi rejeitada em um processo seletivo de emprego. A rejeição certamente trará emoções negativas. Entretanto, para a pessoa, esse fato não pode ser a comprovação de que ela é incompetente e que nunca conseguirá um emprego. Uma maneira mais apropriada de interpretar o acontecimento poderia ser: para as pessoas que faziam a avaliação naquela ocasião, eu não demonstrei ser a pessoa mais qualificada.

Ao atribuirmos um sentido excessivamente negativo a um acontecimento, corremos o risco de nutrir um tipo de ressentimento, que transforma uma dor passageira em uma dor crônica. Nesse

caso, eu acredito que, se eu sofro, alguém deve ser necessariamente culpado por isso e também deve sofrer. Dessa forma, não assumimos responsabilidade sobre nossos próprios pensamentos e emoções, o que é ruim para nosso crescimento individual.

Para lidarmos com a ansiedade e frustração, devemos exercer nossa autoafirmação de maneira sincera conosco. Devemos considerar nossas forças, fraquezas e condições em suas totalidades, e não como moldadas no senso comum. Se o senso comum diz que deveríamos falar inglês fluentemente, por exemplo, e nós nunca estudamos o idioma, devemos aceitar estas condições e agir a partir desta realidade.

8.3 Perfeccionismo

Outro aspecto bastante presente como causa de sofrimento na nossa profissão, e também motivo para longas jornadas de trabalho, é o perfeccionismo. Quando olhamos as empresas, ou a carreira de outras pessoas, de longe, tudo parece perfeito como uma obra de arte. Ao compararmos com nossa própria realidade, onde as coisas acontecem de maneira mais caótica do que gostaríamos, ficamos frustrados. Nós nos sentimos mal, por exemplo, por todos os débitos técnicos que acumulamos em nossos projetos, assim como pelas decisões erradas de negócio.

Entretanto, sabemos que essas empresas também não são perfeitas. Se observarmos por dentro, elas também terão problemas porque foram igualmente construídas por seres humanos que cometem erros. Mesmo cientes disso, muitas vezes realizamos uma autocrítica excessivamente severa sobre nosso próprio trabalho.

Não podemos confundir perfeccionismo com a busca por aprender mais ou dar o nosso melhor, que são pensamentos que nos energizam. O perfeccionismo de que falamos aqui está associado a

sentimentos negativos, como o medo e a culpa. Nas ocasiões em que há o predomínio do medo, acabamos por focar excessivamente no resultado final — por exemplo, na qualidade do software, da apresentação ou mesmo do e-mail que escrevemos — e isso acaba por nos paralisar em vez de nos dar energia para agir.

Um olhar pragmático sobre qualidade

Apesar de o foco excessivo por qualidade nos paralisar, é fato que a qualidade do trabalho importa. Ela está intimamente ligada com a nossa satisfação, que, por sua vez, está conectada ao nosso engajamento e capacidade de produzir — não queremos entregar um trabalho que não nos dá orgulho.

Ao mesmo tempo, nem todo projeto demanda uma qualidade excepcional. Eles poderão ser refatorados no futuro, ou serão descartados, apesar de todos os esforços para sermos efetivos. Eles podem ter servido com o propósito de aprendizado, mas as soluções nunca vão durar para sempre.

Portanto, como podemos balancear as necessidades de negócio com a demanda por qualidade individual? Do ponto de vista de desenvolvimento de produto, uma boa dica é fazer uma avaliação pragmática sobre a vida do usuário antes e depois do projeto que está sendo entregue, ou seja, avaliar o encaixe problema-solução.

Digamos que trabalhamos em um sistema de vendas de ingressos online para eventos e integramos o PIX como um novo método de pagamento — o que era um desejo de um número considerável de usuários. Porém, a integração contém problemas de performance e a usabilidade não está boa. A integração não ficou de um jeito que nos orgulha porque sabemos como poderia ser mais bem implementada. Ela não agrada o ego, mas foi o melhor que foi possível fazer dadas as restrições de tempo.

Se compararmos a vida dos usuários antes e depois do projeto, ela definitivamente está melhor — pessoas que não conseguiam

comprar ingressos por não terem disponível outros métodos de pagamento, agora podem comprá-los com o PIX. Nesse caso, podemos considerar que a entrega é satisfatória, e, dessa maneira, segurar nossos anseios por mais qualidade.

Investigando os pensamentos por trás do perfeccionismo

O olhar pragmático para avaliar situações e desconstruir comportamentos perfeccionistas ajuda, mas pode não ser suficiente. Vários fatores podem levar ao perfeccionismo, e eles podem ter raízes profundas na infância, como o medo de desaprovação e sentimentos de insegurança [8].

Os traços de personalidade construídos ao longo da vida, a partir das nossas experiências, podem ter consequências na relação que construímos com nossa profissão. Um estudo publicado no *The Journal of General Psychology* em 2021 encontrou uma associação entre traços de personalidade como perfeccionismo, narcisismo e uma motivação exagerada para superar outras pessoas, e o *workaholism* (vício em trabalho) entre acadêmicos na Turquia [9].

Por isso, conselhos como "não seja tão perfeccionista", ou "você deve reduzir suas exigências por qualidade", não ajudam. Não há um conjunto pronto de dicas que podem resolver o perfeccionismo, mas existem boas práticas que podem ajudar. Práticas como escrever um diário com afirmações positivas, estar atento e desafiar as conversas internas que contêm autojulgamento podem ser boas estratégias para começar.

Devemos buscar entender se estamos agindo motivados por sentimentos negativos. Nesse caso, uma psicóloga pode nos ajudar a identificar e ressignificar os pensamentos por trás dos comportamentos perfeccionistas, como o medo do fracasso e o desejo excessivo por admiração.

8.4 Conversas com nós mesmos

Uma das maneiras mais importantes para lidarmos com os padrões de pensamentos que nos causam sofrimento é observá-los por meio das conversas que temos com nós mesmos. Ao observarmos com atenção estas conversas, podemos identificar distorções em relação à realidade e, com isso, ressignificar nossas atitudes ao desfazer crenças nocivas.

A abordagem da Comunicação Não Violenta (CNV), que vimos no capítulo sobre conversas difíceis, pode orientar a maneira como realizamos os diálogos internos. A CNV como ferramenta facilitadora para geração de empatia e compaixão também pode ser direcionada para nós mesmos, porque é possível que tenhamos empatia com outras pessoas em situações difíceis ao mesmo tempo em que somos violentos com nós mesmos — por exemplo, quando fazemos afirmações sobre nós carregadas de culpa e ódio.

A relação violenta que construímos conosco muitas vezes vem de uma objetificação de nós. Pensamos que deveríamos funcionar de uma certa maneira, como um objeto que foi concebido para realizar uma determinada função, e que, quando não nos comportamos desta maneira, acreditamos que estamos quebrados e precisamos ser consertados.

Por isso, quando cometemos algum erro, nos recriminamos e pensamos que deveríamos ser punidos. Começamos a nutrir um ódio por nós mesmos, em vez de reconhecer nossas limitações para proporcionar crescimento. Marshall B. Rosenberg, autor da CNV, argumenta que, mesmo que às vezes nós "aprendamos uma lição", devemos nos preocupar com a energia por trás da motivação que nos faz agir. Em suas palavras:

> *"Mesmo quando às vezes 'aprendemos uma lição' com os erros pelos quais nos julgamos com tanta severidade, preocupo-me com a natureza da energia por trás daquele tipo de mudança ou aprendizado. Eu gostaria que a mudança fosse estimulada por um claro desejo de melhorar a nossa vida e a dos outros, em vez de por energias destrutivas como a vergonha ou a culpa."* – Marshall B. Rosenberg [10]

Vergonha e culpa são tipos de ódio de si. Elas representam uma desarmonia entre a maneira como agimos e nossas necessidades de autoafirmação. Entretanto, nesta busca por harmonia entre o que somos e o que fazemos, podemos ressignificar nossas atitudes para que as crenças por trás delas estejam carregadas por respeito e compaixão por nós mesmos, ao invés de culpa ou vergonha.

É natural que às vezes nos arrependamos de coisas que fizemos ou falamos, mas a melhor maneira de aproveitar este arrependimento — e não ficar remoendo o acontecimento — é nos reconectarmos com os sentimentos e necessidades não atendidas que proporcionaram a atitude.

Uma expressão que indica que pode haver sentimentos negativos por trás de nossas ações é quando falamos "eu tenho que". Marshall Rosenberg propõe o exercício de refrasear essas afirmações para: "escolho fazer *, porque quero* ". Essa mudança deve proporcionar reflexões importantes sobre por que fazemos o que fazemos, e pode nos fazer reconhecer um significado maior sobre algo que não gostamos de fazer no dia a dia — ou, ainda, nos fazer mudar nossas atitudes.

Em nossa relação com o trabalho, é normal desejarmos reconhecimento, validação social e prosperidade. Esses desejos estão relacionados com necessidades humanas. Porém, ao reconhecermos estes pensamentos, somos empoderados a sair do

piloto automático e enxergar alternativas na maneira de pensar e agir — ao confrontar nosso próprio ego e vaidade.

Rahaf Harfoush, em seu livro *Hustle and Float: Reclaim Your Creativity and Thrive in a World Obsessed with Work*, sugere algumas reflexões para identificar nossas crenças em relação ao trabalho [11]:

- *Como eu e meus amigos falamos sobre trabalho?*
- *Como eu me sinto trabalhando?*
- *Qual o papel do medo e competição nos meus padrões de trabalho?*
- *Que significado, se houver algum, eu consigo verdadeiramente extrair do trabalho que eu faço? (Uma dica: não há resposta certa ou errada.)*
- *Voltando à idade em que eu escolhi minha carreira, o que motivou a minha escolha? O que eu me imaginei fazendo, desejando e alcançando? Como isso difere de onde estou hoje?*
- *Como eu descrevo minha identidade profissional para mundo exterior? Como isso me serve?*
- *Como eu defino sucesso?*
- *De quais maneiras eu comunico ou sinalizo minha ética de trabalho? Onde eu aprendi isso?*
- *Quais comportamentos são recompensados e respeitados no meu local de trabalho?*

Perguntas como essas podem ser pontos de partida para reflexões importantes, mas, para que elas realmente alcancem este objetivo,

precisamos ser honestos com nós mesmos. Da mesma maneira que, em conversas difíceis com outras pessoas, evitamos falar o que devia ser dito para evitar situações desconfortáveis, podemos ter esse mesmo comportamento com nós mesmos.

8.5 Avaliando o que é essencial

Por fim, uma das habilidades mais importantes que precisamos desenvolver para que possamos manter uma relação sustentável com nossa profissão envolve nossa capacidade de avaliar o que é essencial. Como pessoas apaixonadas por nossa profissão, estamos sempre engajados em muitas iniciativas. Participamos de *meetups*, queremos conhecer as novidades, contribuir para projetos *open-source* e nos manter ativos na comunidade de forma geral.

Muitas vezes, nessa busca por autoafirmação, sentimos que não estamos fazendo o suficiente e começamos a gerenciar o nosso tempo como máquinas. Cada hora do nosso dia ganha uma finalidade, e deixamos que outras pessoas ocupem nossa agenda, incapazes de dizer "não" a compromissos que julgamos serem oportunidades que não podemos deixar passar.

Como Greg McKeown sugere em seu livro *Essencialismo: a disciplinada busca por menos* [12], devemos refletir sobre o que é realmente essencial e falar "não" para o que não for — simples de falar, difícil de colocar em prática. Quanto mais sucesso obtemos em nossa profissão, mais oportunidades recebemos para nos distrair do que realmente importa — e quando as oportunidades chegam, é difícil avaliar o que deve ser rejeitado.

> *"Tudo aceitar, sem discernimento nem preferência, é, de fato, negar a si mesmo, sua sensibilidade, suas prioridades e seus desejos."* – Balthasar Thomass [13]

A negação deve ser vista como um componente importante da autoafirmação. Dessa maneira, por trás de todo "não" deverá haver um "sim" para algo que consideramos mais importante para nós. Não se trata da negação por si só, mas da afirmação de algo mais importante. Essas restrições são o que possibilitam qualquer forma de excelência — elas restringem e intensificam nossa energia para o que consideramos essencial e nos torna capazes de realizar mais. Quando conseguimos colocar isso em prática, nós nos sentimos mais satisfeitos.

Neste capítulo, falamos sobre como padrões de pensamentos e crenças individuais podem ser causas de sofrimento na nossa relação com o trabalho. Apesar de esses padrões e crenças estarem enraizados no espírito do nosso tempo, ainda assim somos capazes de reconhecê-los e modificá-los, atribuindo outro propósito às nossas atitudes.

Com essas reflexões, o que buscamos é a redução do sofrimento, e não sua ausência completa. Para termos realização profissional, precisaremos arriscar a dor em alguma medida. O próprio processo de aprendizado é inerentemente frustrante, porque durante o processo haverá erros e quebras de expectativas. Apesar disso, aprender também gera realização — há satisfação no processo de crescimento e ampliação de nossas capacidades.

Embora o sofrimento esteja presente, não devemos buscá-lo deliberadamente. Isso é o que normalmente nos confunde. O que buscamos é o florescimento de si, e não a dor que pode estar ou não associada ao processo de autoafirmação. A aceitação do sofrimento não deve ser confundida com condescendência por condições extremas de trabalho, injustiças ou comportamentos de autoexploração.

Contudo, precisamos aceitar que quando dizemos "sim" a algo, dizemos "sim" a tudo que vem junto. Quando nos comprometemos a realizar um novo curso, por exemplo, sabemos que precisaremos

dedicar horas de estudo e que, possivelmente, teremos dificuldades e frustrações — significa que, possivelmente, teremos que abrir mão de outras coisas que também valorizamos.

Enquanto o processo de autoafirmação acontece, precisamos estar atentos a nós mesmos. É comum não percebermos que estamos chegando a um esgotamento, porque nem sempre os sinais são claros. Por isso, para prevenir, podemos fazer nossa higiene mental. Assim como escovamos os dentes todos os dias para evitarmos problemas maiores, devemos cuidar da nossa mente dentro de nossa rotina antes de nos reconhecermos em *burnout*. Isso pode significar coisas diferentes para cada um, como ter contato com alguma forma de arte, exercitar-se, praticar ioga, meditar etc. O importante é não esperar que as coisas piorem para reservar tempo para o autocuidado — quando estamos afundados em um ciclo de autoexploração, pode ser difícil reconhecer nossa condição e ter energia para sair dele.

RESUMO

- A síndrome de *burnout* é uma condição de esgotamento associada a uma exposição prolongada de estresse no trabalho;
- Apesar de o *burnout* estar bastante associado ao ambiente — às condições de trabalho — ele também pode acontecer por uma dinâmica de autoexploração. Nesse caso, o próprio indivíduo trabalha incessantemente para realizar uma autoafirmação;
- O perfeccionismo também é causa frequente de sofrimento na vida profissional. A qualidade do trabalho é importante para o engajamento das pessoas, mas precisamos exercitar uma visão pragmática sobre qualidade para avaliar o que merece um esforço adicional ou não;
- Da mesma maneira que precisamos colocar em prática uma comunicação empática com outras pessoas, também precisamos ter empatia nas conversas com nós mesmos. A Comunicação Não Violenta pode auxiliar: ela pode nos ajudar a identificar nossas próprias necessidades e sentimentos, e a esclarecer crenças individuais que podem ser causas de sofrimento.

Referências

[1] MASLACH, Christina; LEITER, Michael P. Defining burnout. *In*: Understanding the burnout experience: recent research and its implications for psychiatry. *World Psychiatry* (online), v. 15, n. 2., jun. 2016. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4911781/#wps20311-sec-0002title. Acesso em 16 set. 2022.

[2] ___. Causes and outcomes. *In*: *Ibidem*.

[3] BURN-OUT an "occupational phenomenon": International Classification of Diseases. *World Health Organization*, 28 maio 2019. Disponível em: https://www.who.int/news/item/28-05-2019-burn-out-an-occupational-phenomenon-international-classification-of-diseases. Acesso em 16 set. 2022.

[4] DEVOPS Enterprise Summit. Understanding Job Burnout. *IT Revolution*, 18 fev. 2019. Disponível em: https://itrevolution.com/understanding-job-burnout-christina-maslach/. Acesso em 16 set. 2022.

[5] HAN, Byung-Chul. *Sociedade do Cansaço*. Rio de Janeiro: Editora Vozes, 2017.

[6] *Ibidem*, p. 115.

[7] VALE, Alisson. *A Fórmula da Eficácia*. Distrito Federal: Software Zen, 2020.

[8] PERFECTIONISM. *GoodTherapy*, 5 nov. 2019. Disponível em: https://www.goodtherapy.org/learn-about-therapy/issues/perfectionism. Acesso em 16 set. 2022.

[9] DOLAN, Eric W. Narcissistic and perfectionistic tendencies are linked to workaholism among academicians, study finds. *PsyPost*, 10 out. 2021. Disponível em: https://www.psypost.org/2021/10/narcissistic-and-perfectionistic-tendencies-are-linked-to-workaholism-among-academicians-study-finds-61949. Acesso em 16 set. 2022.

[10] ROSENBERG, Marshall. *Comunicação não-violenta*: técnicas para aprimorar relacionamentos pessoais e profissionais. São Paulo: Ágora, 2006. p. 181.

[11] HARFOUSH, Rahaf. *Hustle & Float*: Reclaim Your Creativity and Thrive in a World Obsessed with Work *Nova Iorque: Diversion Books, 2019. Edição Kindle, posição 3633. Tradução livre do autor.*

[12] *MCKEOWN, Greg. _Essencialismo*: a disciplinada busca por menos. Rio de Janeiro: Sextante, 2015.

[13] THOMASS, Balthasar. *Afirmar-se com Nietzsche*. Rio de Janeiro: Editora Vozes, 2019. p. 111.

Capítulo 9
Gestão de conhecimento, tempo e energia

Um dos grandes desafios que enfrentamos enquanto desenvolvedores é a gestão individual do conhecimento. Precisamos ser capazes de aprender continuamente, além de suportar uma crescente necessidade por conhecimento do ambiente em que estamos inseridos.

No lado técnico, as ferramentas evoluem — novas versões das linguagens e *frameworks* são lançadas, brechas de segurança são descobertas, as abordagens para resolver problemas mudam, e as plataformas em que nossos softwares operam também se modificam, o que influencia a maneira como desenvolvemos nossas aplicações.

Além do lado técnico, o software está sempre inserido em um ambiente mais abrangente, que é o contexto em que ele opera. Nesse contexto, há outras inúmeras peculiaridades: regras de negócio específicas, leis locais que mudam com o tempo, modelos de negócio diferentes, *stakeholders* com casos de uso específicos do produto etc. A pessoa desenvolvedora, tendo que compreender profundamente todas as nuances do negócio para que possa entregar seus projetos, acaba por se tornar uma especialista no negócio em que o software está inserido.

Neste capítulo, falaremos sobre como lidar com essa demanda incessante por conhecimento. Vamos refletir sobre nosso processo de aprendizado e, também, propor dicas práticas para lidar com o aprendizado continuado. Conectado a isso, também vamos explorar ideias e abordagens para gestão do tempo e energia — o desafio que enfrentamos como trabalhadores do conhecimento se intensifica nas restrições de tempo e energia de que dispomos.

9.1 Preparando-se para uma jornada de aprendizado

Como desenvolvedores de software, a principal habilidade que podemos desenvolver para sermos mais efetivos em nossa profissão é nossa capacidade de aprender. As pessoas que se destacam no nosso mercado são as que, diante de um novo problema, são mais capazes de aprender rapidamente o que for necessário para conseguirem agir na resolução dele de maneira satisfatória.

Passaremos toda a nossa carreira aprendendo para implementar soluções. Em ambientes com tanta complexidade, as pessoas que conseguem lidar melhor com modelos e abstrações têm um diferencial importante, porque elas terão mais facilidade para sintetizar o conhecimento para poderem agir.

Normalmente, os desafios chegam de maneira inesperada — pode ser que não estejamos preparados. Além disso, não há um curso ou passo a passo para superá-los sempre que chegarem. A pessoa desenvolvedora será instigada a formular as perguntas certas e a buscar as informações necessárias para elaborar as soluções que se encaixam com as restrições do momento, sempre colaborando com outras pessoas.

As emoções por trás do aprendizado

Estar nesse cenário de constante aprendizado pode gerar diferentes emoções. Por um lado, isso pode ser causa de ansiedade. Essa sensação de instabilidade pode nos gerar medo e insegurança sobre nossas capacidades, como quando pensamos: "será que serei capaz de resolver este novo problema?".

Por outro lado, a sensação de ter tanto a descobrir também pode ser muito animadora. Quando vemos o progresso que fizemos e percebemos a nossa capacidade de nos adaptarmos, podemos nos

sentir mais encorajados e energizados quando novos desafios chegarem.

O que está por trás do que nos faz agir, se está baseado em uma emoção positiva ou negativa, importa tanto para nossa realização pessoal como para a nossa performance. Como argumentam Tony Schwartz e Jim Loehr, autores do livro *The Power of Full Engagement* [1]:

> *"Para termos o nosso melhor desempenho, devemos acessar emoções agradáveis e positivas: prazer, desafio, aventura e oportunidade. As emoções que surgem da ameaça ou do déficit – medo, frustração, raiva, tristeza – têm uma sensação decididamente tóxica e estão associadas à liberação de hormônios específicos do estresse, principalmente o cortisol. Do nosso ponto de vista, a inteligência emocional é simplesmente a capacidade de gerenciar as emoções habilmente a serviço da alta energia positiva e do engajamento total*".

Isso não significa negar as emoções negativas. Elas são importantes porque nos fazem agir com urgência quando necessário. Também podem proporcionar investigações mais profundas sobre nós mesmos e o ambiente em que estamos inseridos. Entretanto, se queremos um ritmo de alta performance sustentável, não podemos basear nosso impulso para agir em emoções negativas a todo momento.

Para não deixarmos que as emoções negativas tomem conta de nossos pensamentos, podemos fazer duas coisas: primeiro, ampliar nossa capacidade de enfrentar situações difíceis de maneira positiva; e depois, recuperar nossa energia emocional com regularidade. Para ampliar nossa capacidade de lidar com situações difíceis, temos que expandir competências como autoconfiança, autoestima, autocontrole e paciência, e podemos fazer isso ao exercitar essas habilidades. Por exemplo, uma das estratégias para lidar com o

medo, pela abordagem da Terapia Cognitiva-Comportamental, é expor o indivíduo às causas deste medo de maneira controlada, para que a pessoa consiga aprender a superá-lo [2].

O modelo ABC proposto pela TCC ilustra alguns dos seus principais conceitos e pode ser bastante útil para proporcionar mudanças na maneira como interpretamos situações. Com isso, podemos transformar a nossa energia emocional. A ideia é que nós identifiquemos três fatores relacionados a um acontecimento, que são:

- As consequências — as emoções que sentimos e as ações que tomamos que não nos agradam;
- O evento de ativação — o gatilho que proporcionou a ativação de sentimentos e comportamentos nocivos;
- As crenças — a interpretação que tivemos sobre o evento de ativação.

Ao realizar esse exercício, nos tornamos capazes de identificar distorções cognitivas, como: catastrofização (pensar que o pior de uma situação vai acontecer), adivinhação (tentar prever o futuro), filtro mental (focar apenas nos aspectos negativos de um acontecimento), leitura mental (tirar conclusões precipitadas sobre o que os outros estão pensando) e rotulação (colocar rótulos em nós mesmos ou em outras pessoas). Com isso, desafiamos os nossos pensamentos e somos capazes de criar alternativas.

Além de exercitarmos nossa capacidade de lidar com situações difíceis, também precisamos recuperar nossa energia emocional com frequência. As pessoas têm maneiras diferentes de se recuperar emocionalmente — por exemplo, algumas podem precisar de solidão para recuperar as energias e outras, de companhia.

Para recuperar nossa energia emocional, é importante encontrarmos as atividades que realizamos pelo puro prazer de fazê-las. Estamos condicionados a estarmos sempre ativos e produtivos, e a simples ideia de relaxar e engajar em uma atividade não necessariamente

produtiva pode causar desconforto. Entretanto, trabalhar as emoções por trás das nossas ações é crucial para a sustentabilidade do nosso trabalho ao longo do tempo — especialmente na condição de constante aprendizado, que requer muita energia e intenção. Estar atento a elas deve contribuir para a nossa performance e satisfação.

A diferença entre educação e aprendizado

Quando falamos sobre aprendizado, é importante diferenciarmos o *educar* do *aprender*. Por um longo período da nossa infância, e até a uma parte da vida adulta, fomos levados a salas de aula onde recebemos instruções sobre o que aprender e como seríamos avaliados. Esse é o modelo de educação empurrada que estamos bastante familiarizados. O aluno é tratado, e também se comporta, como objeto recebedor de conhecimento. É um modelo caracterizado pela ausência de propósito — há uma desmotivação generalizada porque a maioria das pessoas não vê sentido no estudo, além da própria conclusão do curso e obtenção do certificado.

Quando chegamos à vida adulta, nós nos vemos em um cenário completamente distinto. Diferente do que aconteceu por toda a vida até então, vemo-nos com a necessidade de fazer nossas próprias escolhas. Não há instruções a serem seguidas para que avancemos para a próxima fase, nem garantias sobre os resultados. Você finalmente é desafiado a ser agente da sua própria vida.

Algumas pessoas sentem-se encorajadas nessa nova condição, porque sentem que finalmente podem se afirmar como as pessoas que são. Outras não lidam muito bem, e acumulam ressentimento porque se sentem injustiçadas.

Formação na infância	Construção na fase adulta
O indivíduo como objeto (mim)	O indivíduo como agente (eu)

Formação na infância	Construção na fase adulta
Quem eu preciso ser	Quem eu sou
Uma projeção do social	Uma harmonização com o social.

Extraído do curso Wise & Effective da Software Zen (T1E02: A origem do "eu" que aprende).

Temos que reconhecer que a possibilidade de fazermos nossas próprias escolhas é boa. Entretanto, nem sempre estamos preparados para isso. Para nossa saúde mental, é importante que o aprendizado aconteça de maneira autodirigida — não no sentido de que você deva aprender sozinho, sem ajuda de ninguém, mas para que você seja agente da atividade. Isso significa que você puxou a atividade para si, criou a necessidade e deu sentido a ela. A conexão disso com nossa saúde mental existe porque o aprendizado está intimamente conectado com a nossa construção como indivíduos e com a coerência que há entre o que somos e o que fazemos.

Dessa maneira, quando paramos de olhar para a educação como formação e passamos a olhar para o aprendizado como construção, deixamos de nos posicionar como objetos recebedores de conhecimento e nos colocamos como agentes. Essa mudança de interpretação, que acontece no âmbito individual, é primordial para que possamos trilhar uma carreira sustentável e com mais satisfação.

Aprendizados funcionais e disfuncionais

Ao longo de nossas carreiras, nem tudo que aprendemos é bom. Imagine, por exemplo, a seguinte situação. Você, em uma posição de líder técnico, foi à sua primeira reunião com um gerente de produtos da empresa em que trabalha para apresentar o progresso das últimas duas semanas da equipe que você gerencia.

A equipe de fato fez um bom trabalho e avançou sobre várias perspectivas — entregou funcionalidades que estavam planejadas, resolveu bugs e débitos técnicos, e conseguiu feedbacks valiosos de usuários por meio de dados que foram analisados para a próxima iteração. Entretanto, o gerente de projetos fica irritado porque 100% das tarefas não foram concluídas. Ele continua pedindo funcionalidades que você entende que não contribuem para o sucesso do projeto, em termos de resultados de negócio para a empresa. No fim daquela experiência, você "aprende" que gerentes de produtos não entendem nada de desenvolvimento de software. Na próxima vez, já que você "aprendeu" que gerentes só se importam com tarefas concluídas ou não concluídas, você vai agir diferente no momento de orçar tarefas e preparar-se para as reuniões — os resultados não importam, apenas a posição dos *cards* no board que interessam. Evidentemente, esse é um aprendizado disfuncional, mas por quê?

Podemos considerar aprendizados disfuncionais aqueles que nos afastam do que somos e que nos tiram a potência para agir de acordo com o que acreditamos e com o nosso propósito. No exemplo anterior, o líder técnico acredita que faz parte do seu papel estar próximo dos resultados de negócio, mas, com sua experiência, ele "aprendeu" que deve agir como um entregador de tarefas. Esse é um aprendizado que afasta a pessoa de quem ela é do que ela acredita.

Especialmente depois de uma experiência profissional ruim quando havia uma grande expectativa, é comum nos colocarmos em uma condição propícia para internalizarmos aprendizados disfuncionais. Por exemplo, pense em uma pessoa desenvolvedora que desejava muito trabalhar em uma determinada empresa, porque parecia ser um lugar com uma cultura que valorizava o respeito e que colocava conceitos ágeis verdadeiramente em prática. Ela consegue isso, mas logo em seguida percebe que o ambiente não era nada do que ela tinha imaginado, e acaba por ser desligada da empresa três meses depois. Essa pessoa pode pensar que não existem empresas que

prezam por respeito e que os conceitos ágeis, na verdade, não funcionam na prática. Mas, esse aprendizado não reflete a realidade, e ele tira a força de agir da pessoa — ou seja, agora ela tem menos energia para buscar um ambiente mais respeitoso e com maior potencial de crescimento.

Por outro lado, aprendizados funcionais são o oposto. Eles ampliam nossa capacidade para agirmos em direção ao nosso propósito. Como Alisson Vale articula na primeira temporada do curso *Wise & Effective*, da Software Zen, aprender com a experiência é selecionar e extrair o que tem potencial de construí-lo(a) como pessoa na direção em que você deseja.

Resgatando o exemplo elaborado anteriormente, o líder técnico poderia fazer uma interpretação diferente. Ele poderia aprender a extrair daquela situação o que está passando despercebido em termos de necessidades não contempladas do gerente de produto. Será que houve uma má priorização das atividades? Será que há uma pressão externa da qual não tenho conhecimento? Será que houve um entendimento errado dos resultados de negócio esperados? Será que esta pessoa está passando por um dia difícil?

O aprendizado por meio da experiência, e essa avaliação e seleção dos aprendizados funcionais e disfuncionais, são algo bastante desafiador. Isso envolve nos colocarmos em posições de vulnerabilidade, o que causa certa ansiedade — vamos cometer erros, os resultados podem não ser o que desejávamos e teremos que lidar com estas frustrações ao longo do processo. Entretanto, se houver um significado maior sobre a atividade, bem como uma energia positiva, poderemos ter a coragem necessária para assumir riscos, atentos a nossas limitações. Reconhecer nossas imperfeições e limitações significa nos reconciliarmos com nossa própria humanidade.

Construindo seu próprio sistema desenvolvedor de conhecimento

Além de nos colocarmos como agentes e de estarmos atentos ao propósito e significado que atribuímos ao nosso processo de aprendizado, para lidarmos com o desafio de estar em constante evolução, é muito útil desenvolvermos algum sistema de apoio para gerenciarmos novas informações, conhecimentos e ideias. Em outras palavras, podemos ser mais eficientes se nos apoiarmos em um sistema que nos ajude a ampliar nosso domínio sobre o processo de aprendizagem.

Nesse momento, perguntas como "o que aprender?", "qual é a linguagem certa?", "qual o melhor livro?" ou "qual a melhor abordagem de gestão?" não são tão importantes. O que buscamos é expandir nossa capacidade de aprender sobre os conteúdos que consumimos e, especialmente, sobre as experiências que vivemos.

Para esta seção, vamos buscar inspiração no sistema **Zettelkasten**. Como muitas das recomendações apresentadas ao longo desse livro, receba-a como uma inspiração para você construir seu próprio sistema de gestão de conhecimento, e não como um conjunto de regras que devem ser seguidas.

O Zettelkasten é um sistema de anotações desenvolvido pelo cientista social alemão Niklas Luhmann. Nascido em 1927, Luhmann é ainda hoje uma das maiores referências na área de ciências sociais. Além de conhecido pelo trabalho brilhante que desenvolveu na área de sistemas sociais, Luhmann também ficou famoso pela sua impressionante produtividade — publicou cerca de 50 livros e mais de 600 artigos ao longo da sua vida [3].

Por ser um cientista com uma capacidade excepcional de produzir e sintetizar ideias complexas, o seu sistema de anotações gerou muito interesse. De maneira sucinta, trata-se de um sistema simples de anotações, onde cada uma das notas pode referenciar outras notas, formando uma rede de informações e ideias.

Esse sistema não tem o propósito de ser apenas um acumulador de informações. Quando criamos uma nota sobre algo que estamos

estudando, organizando nossos pensamentos com nossas próprias palavras e transformando-os em frases completas, nos forçamos a processar estas informações com mais profundidade. Quando elas ficam disponíveis em uma rede de anotações mais abrangente, fica mais fácil fazermos conexões e produzirmos *insights*.

> *"Se você não lê apenas por entretenimento, ler sem tomar notas é, em grande parte, um desperdício de tempo. Muito pouco do que lemos é absorvido se nós não exercitamos de forma prática, se não elaboramos ideias novas em cima do conteúdo ou se nós não o relacionamos com outros conteúdos. Sem um sistema de prática deliberada ou de processamento do conteúdo, você acaba por ser apenas um colecionador de informações."* – Alan Fachini [4]

Niklas Luhmann indexava e organizava suas anotações manualmente. Hoje em dia, temos softwares que podem fazer este papel de maneira mais eficaz, como o Roam Research [5], o Obsidian [6] ou mesmo planilhas.

Colocar o novo conhecimento adquirido em prática é parte crucial para o processo de aprendizagem. Quando estamos aprendendo um novo idioma, por exemplo, ampliamos o nosso vocabulário à medida que começamos a utilizar as novas palavras descobertas na prática. Você não pega um dicionário e decora todas as palavras contidas nele, porque essas informações, desconectadas de qualquer experiência prévia, não serão lembradas. As palavras só se tornam acessíveis em nossas mentes quando exercitamos o seu uso nos contextos apropriados.

Da mesma maneira, se estamos aprendendo uma nova linguagem de programação, devemos proporcionar experiências de desenvolvimento com aquela linguagem para que possamos internalizar os novos conceitos. Se quisermos melhorar a nossa habilidade de implementar testes e técnicas de *mock*, devemos

escrever mais testes. Se, como gerentes, temos dificuldade em fazer boas conversas um a um, temos que fazer isso mais vezes.

9.2 Gerenciando tempo e energia

Com tantas responsabilidades e tanta necessidade de aprender coisas novas, tentamos organizar a nossa agenda como se fôssemos máquinas. Acordamos mais cedo para conseguir trabalhar sem interrupção. Almoçamos rápido para poder voltar antes ao trabalho. Usamos nossas noites para cursos e projetos paralelos. Mesmo quando estamos cansados, nossos compromissos opõem-se ao que nosso corpo diz — nos tornamos obcecados pelo tempo.

> *"Disciplinamos nossas vidas pelo tempo no relógio. Nossos trabalhos e salários são determinados por ele, e muitas vezes nosso 'tempo livre' também é rigidamente administrado por ele. De um modo geral, até nossas funções corporais são reguladas pelo relógio: geralmente comemos nossas refeições em horários apropriados ao invés de quando estamos com fome; vamos dormir em horários apropriados ao invés de quando estamos cansados e atribuímos mais importância aos tons impressionantes de um alarme de relógio do que o aparente nascer do Sol no centro do nosso sistema solar."* – Joe Zadeh [7]

Costumamos tratar o tempo como algo absolutamente preciso e real. Entretanto, como Joe Zadeh escreve eu seu artigo intitulado *The Tyranny Of Time*, o tempo que marcamos em nosso relógio não passa de uma criação humana. O planeta não rotaciona em seu próprio eixo em exatamente 24 horas, assim como ele não orbita o Sol em exatamente 365 dias — "*a perfeição é uma criação humana; a natureza é irregular*" [8]. Estamos constantemente fazendo correções no tempo, ou alterando-o por meio de decisões políticas,

como quando ajustamos o relógio para aproveitar melhor o dia nos horários de verão.

A criação do tempo tem sido definitivamente útil na maneira como estabelecemos nossas relações sociais. Com ele, podemos fazer acordos e planejar nossas vidas. Mas, ao mesmo tempo, passamos a interpretar o tempo apenas como uma commodity que pode ser usada ou desperdiçada. Esta relação com o tempo pode ser bastante prejudicial, porque "*quanto mais nos sincronizamos com o tempo no relógio, mais nos dessincronizamos com nosso próprio corpo e o mundo ao nosso redor*" [9].

Quando nos empenhamos em alguma atividade, não nos dispomos apenas de tempo, mas também de energia. Ao nos engajarmos em uma atividade às 9h, a quantidade e qualidade da energia de que dispomos é bem diferente da de que se estivéssemos realizando esta mesma atividade às 21h. A quantidade e qualidade de nossa energia varia ao longo do dia, e precisamos aprender a gerenciá-la ao nosso favor.

Gerenciando nossa energia em favor do engajamento

Como não podemos ver a nossa energia, é comum agirmos como se ela fosse ilimitada. Apesar de não podermos medi-la da mesma maneira que fazemos com o tempo, podemos sim gerenciá-la a nosso favor. Esta é a alternativa para a vida completamente voltada à gestão do tempo. Em vez de organizarmos nossas vidas como se fôssemos máquinas, aceitamos o fato de que temos uma quantidade limitada de energia, e que precisamos recuperá-la com regularidade.

> *"As formas de viver que são mais ricas, felizes e produtivas são caracterizadas pela habilidade de engajamento completo no desafio que se apresenta, seguido de um desengajamento periódico e pela busca de renovação."* – Jim Loehr e Tony Schwartz [10]

Quando organizamos nossa vida apenas pelo tempo, consideramos que nossa energia está disponível ao longo do dia de maneira linear. Entretanto, não é assim que realmente vivemos. Nossa energia oscila ao longo do dia, e, pensando nisso, podemos intencionalmente estabelecer um balanço entre estresse e descanso para poder aproveitar o melhor de nós mesmos.

Muitas das ideias contidas nesta seção foram inspiradas no livro *The Power of Full Engagement*, de Jim Loehr e Tony Schwartz. No livro, os autores relatam as abordagens e resultados obtidos com seus clientes quando estes precisavam melhorar suas performances ou se recuperarem de períodos difíceis. A ideia central da abordagem está na gestão da energia e na disciplina pela recuperação da energia.

Em um estudo realizado com tenistas profissionais, eles descobriram que o que diferenciava os atletas que estavam no topo do ranking do restante era a maneira como eles eram capazes de recuperar energia durante a partida — no curto período entre os pontos e sets. Nesses intervalos, os atletas de ponta colocavam em prática rituais que os faziam amenizar a frequência cardíaca e recuperar a energia e concentração para o ponto seguinte.

Jim e Tony descobriram que o ponto de alavancagem para seus clientes não estava no treino técnico — seus clientes sabiam exatamente como colocar em prática os movimentos e técnicas necessárias para vencer uma partida —, mas sim na gestão de energia nos treinos e durante as partidas. O controle da respiração e a capacidade dos atletas de lidarem com frustrações contava mais naquele momento.

Apesar de eles terem iniciado seus trabalhos com tenistas profissionais, a ideia central foi experimentada com profissionais das mais diversas áreas. Em todos os casos, Schwartz e Loehr buscaram maneiras de expandir e recuperar o que eles chamaram de quatro tipos de energia: física, emocional, mental e espiritual.

Nossos quatro tipos de energia

Nesta seção, vamos destrinchar o que são os quatro tipos de energia descritos por Schwartz e Loehr e como podemos conectar estes conceitos com o nosso dia a dia como pessoas desenvolvedoras de software.

O primeiro tipo de energia que eles descrevem é a **energia física**. Apesar de exercermos uma atividade dita como sedentária, não significa que não precisamos dispor de energia física para desempenhá-la — nossa capacidade de realizar tarefas ao longo do dia não é infinita só porque estamos sentados.

Muitas vezes, quando sentimos falta de energia durante o dia, compensamos tomando bebidas estimulantes como o café. Mais tarde, quando não conseguimos relaxar no final do dia, é comum recorrermos ao álcool. Entretanto, manter este ciclo por um longo prazo é prejudicial para nossa saúde. Sintomas como hiperatividade, agressividade, impaciência, irritabilidade e raiva se tornarão mais frequentes enquanto não respeitarmos o nosso ciclo de recuperação de energia.

Fazer pausas durante o dia, depois de uma ou duas horas de trabalho concentrado, é muito benéfico para nosso corpo e mente. Usar estas pequenas pausas para fazer caminhadas leves pode contribuir muito para redução de estresse e recarregar nossas energias e capacidade de concentração.

Evidentemente, para a recuperação de nossa energia física, uma boa noite de sono também é fundamental. Invista nas boas práticas para dormir bem, como: evitar café ou bebidas alcoólicas à noite; reduzir seu contato com telas por pelo menos 30 minutos antes de ir para a cama; preparar um ambiente escuro e silencioso; e fazer seu ritual de desligamento.

O segundo tipo de energia é a **energia emocional**. Diferente da energia física, onde medimos sua quantidade, na energia emocional nós avaliamos sua qualidade — se é positiva ou negativa. Assim como a energia física, sua recuperação também é importante.

> *"Em termos práticos, os 'músculos-chaves' ou competências que nos abastecem com energia positiva são autoconfiança, autocontrole (autorregulação), habilidades sociais (eficácia interpessoal) e empatia. 'Músculos' menores de apoio incluem paciência, abertura, confiança e prazer."* – Jim Loehr e Tony Schwartz [11]

Em seções anteriores falamos sobre como a TCC pode nos auxiliar a construir estes "músculos" aos quais Schwartz e Loehr se referem. Além disso, técnicas que exercitam a atenção plena por meio da respiração também podem ajudar bastante. Práticas de meditação focadas na respiração possuem benefícios comprovados em termos de: aumento de nossa capacidade de concentração; redução da divagação mental; controle de nossa reatividade emocional, entre outros [12].

Refletir sobre a energia emocional que está por trás da nossa motivação para agir é fundamental. Agir baseado apenas em emoções negativas, como raiva e medo, é menos eficiente — nossa capacidade de concentração será menor, assim como nosso desempenho —, além de não ser sustentável no longo prazo.

O terceiro tipo de energia é a **energia mental**. Ela está relacionada a nossa criatividade e capacidade de resolver problemas. Ela também depende das duas energias descritas anteriormente — se estamos fisicamente cansados, não somos capazes de focar, e se estamos emocionalmente ansiosos ou com medo, não conseguimos ser criativos.

Assim como os demais tipos de energia, a recuperação intermitente é um aspecto importante. A oscilação entre concentração e relaxamento é importante para a criatividade e para a nossa capacidade de avaliar e julgar. No nosso dia a dia como desenvolvedores, estamos a todo momento precisando fazer avaliações e ser criativos. Por exemplo, nos momentos em que nos

perguntamos "esta é a maneira mais simples de implementar esta solução?", ou "existe alguma brecha de segurança nesta funcionalidade?", ou, ainda, "há maneiras melhores de organizar este código para que promova uma melhor manutenibilidade?", estamos demandando capacidade de avaliação e julgamento.

> *"Um bom plano é de vez em quando sair e relaxar um pouco (...). Quando você volta para o trabalho, seu julgamento será mais assertivo, uma vez que permanecer constantemente trabalhando fará você perder poder de julgamento."* – Leonardo da Vinci, como citado em *The Power of Full Engagement*, de Jim Loehr e Tony Schwartz

Para que possamos ter boa concentração e foco, precisamos ser capazes de priorizar nossas atividades de maneira apropriada. Para isso, algumas boas práticas incluem:

- Avaliar quais atividades são urgentes e quais são importantes;
- Ter as informações quanto a restrições associadas às tarefas — prazos, requisitos etc.;
- Organizar um ambiente de trabalho livre de distrações — desligar notificações do celular, fechar e-mail etc.;
- Saber delegar atividades que você não pode absorver.

Por fim, temos a **energia espiritual**. Apesar do nome, ela não tem um sentido religioso. Schwartz e Loehr a colocam como algo que está relacionado ao propósito que está por trás das atividades em que despendemos nossa energia. Ela está conectada com o nosso conjunto de valores e com o sentido que atribuímos às ações que estão além do nosso interesse próprio.

Resumo

- Ao longo de nossa carreira como desenvolvedores, precisaremos sempre lidar com o desafio do aprendizado. Nossa habilidade em lidar com estes desafios influenciará enormemente a nossa eficácia como desenvolvedores, assim como a nossa satisfação;
- Há uma diferença entre educação e aprendizado. Na educação, nós nos colocamos como objetos recebedores de conhecimento, enquanto no aprendizado nos colocamos como agentes da atividade. Para nossa saúde mental, é importante que o aprendizado aconteça de maneira autodirigida para que sejamos responsáveis por atribuir sentido à atividade;
- É importante sermos capazes de diferenciar aprendizados funcionais e disfuncionais. Aprendizados funcionais ampliam nossa capacidade de agir de acordo com quem somos, e aprendizados disfuncionais limitam nosso potencial de agir;
- Como trabalhadores do conhecimento, podemos nos beneficiar de sistemas que apoiam a gestão de conhecimento pessoal — como o sistema de anotações Zettelkasten e softwares como Obsidian e Roam Research, que facilitam o acesso e conexão de ideias;
- Em vez de focarmos completamente na gestão do tempo, devemos prestar mais atenção a como gerenciamos nossa energia — como expandimos e recuperamos nossa energia física, emocional, mental e espiritual;
- Reconhecer nossas limitações é nos reconciliarmos com nossa própria humanidade.

Referências

[1] LOEHR, Jim. SCHWARTZ, Tony. *The Power of Full Engagement*. Nova Iorque: Free Press, 2003. p. 72. Tradução livre do autor.

[2] BRANCH, Rhena. WILSON, Rob. *Terapia Cognitivo-Comportamental Para Leigos — Edição de Bolso.* Rio de Janeiro: Alta Books, 2019. p. 93.

[3] SASCHA. Luhmann's Zettelkasten. *In*: Introduction to the Zettelkasten Method. *Zettelkasten* (website). Alemanha, 27 out. 2020. Disponível em: https://zettelkasten.de/introduction/#luhmann-s-zettelkasten. Acesso em 16 set. 2022.

[4] FACHINI, Alan R. Desenvolvendo conhecimento. *Magrathea*, 7 ago. 2020. Disponível em: https://blog.magrathealabs.com/desenvolvendo-conhecimento-5324537df36b.

[5] ROAM RESEARCH. (Site Institucional). Disponível em: https://roamresearch.com/.

[6] OBSIDIAN. (Site Institucional). Disponível em: https://obsidian.md/.

[7] ZADEH, Joe. The Tyranny Of Time. *Noema Magazine* (online). Los Angeles: Berggruen Institute, 3 jun. 2021. Disponível em: https://www.noemamag.com/the-tyranny-of-time/. Acesso em 16 set. 2022.

[8] *Ibidem*.
[9] *Ibidem*.
[10] LOEHR, Jim. SCHWARTZ, Tony. *The Power of Full Engagement*. Nova Iorque: Free Press, 2003. p. 12.
[11] *Ibidem*. p. 72. Tradução livre do autor.
[12] TYRRELL, Fiona. *THE YOGI masters were right – breathing exercises can sharpen your mind*. Trinity College Dublin. Irlanda: The University of Dublin, 10 maio 2018. Disponível em: https://www.tcd.ie/news_events/articles/the-yogi-masters-were-right-breathing-exercises-can-sharpen-your-mind/. Acesso em 16 set. 2022.

CONSIDERAÇÕES FINAIS

CAPÍTULO 10
Assumindo nossa responsabilidade

Falar sobre sustentabilidade humana, saúde mental e a nossa relação com o trabalho é uma tarefa difícil. Existem muitos fatores que precisam ser considerados, e quando nos aprofundamos sobre eles, esses fatores podem parecer contraditórios. No decorrer deste livro exploramos diversos temas, separando-os em três grandes escopos: o time e suas formas de trabalhar; a empresa e sua cultura; e o indivíduo e suas crenças.

Na primeira parte, vimos como uma gestão inadequada de projetos de software pode comprometer a sustentabilidade do trabalho. Quando não somos capazes de gerenciar de maneira apropriada as incertezas intrínsecas da construção de produtos de software, isso acaba por destruir a confiança das relações e criar um ambiente tóxico para as pessoas.

Na segunda parte, aprofundamo-nos na cultura das empresas. Em muitas organizações, há expectativas irreais sobre disponibilidade que levam as pessoas a trabalharem longas jornadas, porque isso faz parte das regras não escritas da companhia. Mesmo quando a empresa possui políticas e benefícios que permitem que os colaboradores passem mais tempo com suas famílias, há algo no ar que impede que essas pessoas usufruam disso. Disponibilizar essas políticas é importante, mas não é suficiente — a cultura das empresas deve abraçar um ritmo de trabalho sustentável.

Além dos fatores organizacionais, temos os aspectos individuais. As nossas crenças pessoais, assim como a maneira como gerenciamos nosso tempo e energia, influenciam muito nossa capacidade de manter e melhorar nossa performance ao longo do tempo. Em nossa busca por autoafirmação, muitas vezes caímos em ciclos de

autoagressão que nos impedem de nos desconectarmos do trabalho, e esses comportamentos possuem consequências severas na nossa saúde mental e física.

Apesar de não haver um conjunto de regras prontas que resolvam todos esses problemas, temos boas práticas que podem ser seguidas e que podem ajudar bastante. Podemos refinar nossas práticas de gestão; fortalecer a confiança entre nossas relações de trabalho; reconsiderar a cultura de nossas empresas e modificar nossas próprias crenças individuais.

Mais importante, devemos assumir responsabilidade sobre as consequências de nossa maneira de agir. É nossa responsabilidade colaborar para que possamos construir um ambiente que permite às pessoas trabalharem por anos sem que sofram consequências severas na saúde por decorrência do trabalho.

Desenvolver software pode ser extremamente gratificante. Com algumas mudanças na maneira como nos relacionamos com nossa profissão, podemos colocar nosso trabalho como algo construtor de saúde e bem-estar, e não apenas como causa de estresse crônico, ansiedade e *burnout*.

Obrigado por ter lido este livro. Espero que ele tenha ajudado você a construir uma relação mais saudável e gratificante com nossa profissão.

Capítulo 11
Construindo um plano de ação

A proposta deste último capítulo é organizar uma série de perguntas que podem guiar a construção de um plano de ação para você ou para sua empresa. Seja curioso e responda com sinceridade. Com essas reflexões, um plano de ação deve naturalmente emergir, e você pode usar os capítulos anteriores do livro para buscar mais insights sobre temas específicos.

11.1 Avaliando nossa maneira de trabalhar

1. Como pessoa desenvolvedora, como você descreveria suas responsabilidades? Depois de responder, como você se sente sobre a descrição que elaborou? Ela está de acordo com a maneira como você trabalha atualmente em sua equipe?
2. Como você descreveria o propósito do projeto ou produto que você desenvolve hoje? Por que você acredita que esse propósito é (ou não é) relevante?
3. Como se dá a negociação de prazos de projetos dentro da sua empresa? Você pensa que ela acontece de maneira justa ou injusta? Por quê?
4. Como a equipe controla a quantidade de trabalho em andamento? Você se sente estressado por ter de lidar com vários assuntos ao mesmo tempo?
5. Na última vez em que houve um atraso em um projeto, como a equipe se comportou? O que você faria diferente naquele momento?
6. Como a equipe planeja a entrega de um projeto grande (que pode levar mais de um mês)? Como a equipe gerencia os pontos de maior incerteza dos projetos em momentos em que é pressionada a estabelecer compromissos?

7. Você sente que a equipe frequentemente supera seus desafios com força bruta (estende a jornada de trabalho)? Por que você pensa que isso acontece?
8. A equipe consegue entregar software de maneira contínua e com iterações curtas? Estas iterações ficam apenas na equipe de desenvolvimento ou se estendem às outras áreas da companhia e demais *stakeholders* dos projetos? Na sua visão, qual a razão para isso acontecer (ou deixar de acontecer)?
9. A equipe possui o ferramental necessário para entregar software de qualidade continuamente? Ela recebe o apoio das demais áreas da companhia para manter estas ferramentas?
10. Você considera que sua equipe possui uma grande rotatividade? Por que você acredita nisso e como isso impacta seu trabalho?
11. Como você avalia o sentimento de pertencimento das pessoas na equipe? Você sente que a sua opinião e a de outras pessoas são ouvidas?
12. Como você avalia a segurança psicológica da sua equipe? Há um ambiente convidativo à participação? Você sente que pode se colocar em posições de vulnerabilidade, expor opiniões e discordar de outras pessoas?
13. Quando alguma falha é comunicada, a equipe responde de maneira produtiva ou age em busca de culpados? Como você e a equipe se comportaram na última vez que tiveram um incidente grave?
14. Como você avalia a carga cognitiva necessária para lidar com o software que você desenvolve? Ela está suportável? Quanto tempo um novo desenvolvedor que entra na equipe precisa para começar a ser produtivo?
15. Você sente que a equipe consegue entregar software com autonomia, ou há muito *hand off* no processo de desenvolvimento?

11.2 Avaliando a cultura da empresa

1. Quando você pensa sobre a cultura da sua empresa, quais sentimentos surgem em você? As lideranças da companhia são pessoas que colocam o trabalho à frente de valores mais importantes, como a família e a saúde?
2. Você acredita que o tempo das pessoas é respeitado no dia a dia? Você considera difícil manter-se concentrado por uma ou duas horas sem interrupção enquanto trabalha? Se sim, por que você acha que isso acontece?
3. Você sente que pode usufruir dos benefícios da companhia, especialmente quando envolvem dias de folga?
4. Avalie sua agenda do último mês — você se sente satisfeito com ela? O que você adicionaria ou removeria dela? Você sente que uma boa parte do seu tempo é tomado por seus colegas de trabalho sem necessidade? Você consegue separar períodos de descanso para recuperação durante o dia?
5. Você sente que consegue ter conversas difíceis com seus pares? Qual foi a última vez em que isso aconteceu? Como a conversa sucedeu e o que você mudaria da sua postura? Você sentiu que pôde falar plenamente o que sentia e pensava?
6. Como a organização lida com falhas? Da última vez em que você precisou reportar um erro, como se sentiu e como foi a reação da sua liderança e de seus pares?
7. Dentro do seu escopo de trabalho, você sente que tem autonomia para tomar decisões que determinam o que fazer e como fazer?
8. Você pensa ser reconhecido e recompensado de maneira justa por sua performance? Você sente que pode compartilhar esta opinião com sua liderança? Por quê?

11.3 Avaliando suas próprias crenças e gestão individual

1. Quando você conhece uma pessoa, como você se apresenta? Em qual medida sua profissão determina quem é você?
2. Quão difícil é para você desligar os pensamentos relacionados ao trabalho após o expediente? Como isso afeta a sua vida e a qualidade do seu tempo com sua família e amigos? No fim do dia, você sente que tem energia para outras atividades que não estão relacionadas a algum tipo de trabalho produtivo?
3. Como você fala sobre seu trabalho para seus amigos?
4. Qual é o papel do medo e da competição no seu impulso para agir?
5. Você sente que está constantemente em busca do que é tendência no desenvolvimento de software, ou sente que consegue direcionar o caminho que deseja seguir? Você sente que está no controle da sua jornada de aprendizado?
6. Quão importante para sua própria satisfação você considera a qualidade do trabalho que entrega? Por quê?
7. Você se considera uma pessoa perfeccionista? Como isso afeta seu trabalho? Quais são as motivações por trás desse perfeccionismo?
8. Você consegue adicionar intervalos regulares de descanso em sua rotina de trabalho?
9. Como você lida com a demanda constante por conhecimento? É algo que o(a) energiza ou algo que lhe causa ansiedade e medo?